Sinclair QL World

9 770951 933023 01

Volume 3 issue 1 price £2.50

QVME

Enhanced QL on the Atari Ste from Merz

QVME

?

TroubleShooter

Founts Built in or Software

QL Scene

New Developments on QXL and Gold Card

DIGITAL PRECISION

THE POWER BEHIND THE BUTTON

1984-1994

THE BEST SOFTWARE FOR THE BEST COMPUTER

A PRIDE IN ITS LOGIC

NO IDEALISTIC PRIG

SPIRITED LOGICIAN

DIGITAL PRECISION

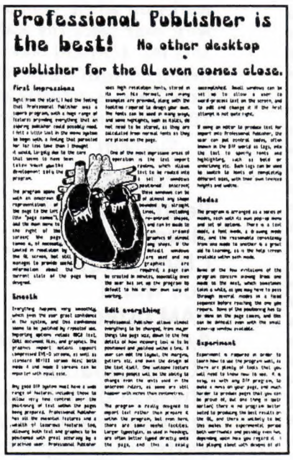
A LOGIC IN ITS PRIDE

PERFECTION SPECIAL EDITION

POWER

PERFECTION SPECIAL EDITION has 253 (two hundred and fifty three) direct/menu commands (not counting options in sub-menus), plus 32 special characters (like Bold on) that can be inserted 'directly' plus intelligent (and now excellently documented) macros. Comparisons with other word processors on the subject of power are hence quite unnecessary.

EASE OF USE



Independent reports, customer feedback and published reviews (of its less able but still excellent predecessor, PERFECTION) leave one in no doubt as to which word processor is friendliest – PERFECTION SPECIAL EDITION, with its intuitive, silky handling. Uniquely, it has two operating modes, with both menus (visible or invisible – they even look like Quill's) and direct commands (for when you familiarise yourself with the system). Uniquely, both modes are 're-entrant' (so you can use any menu option or direct command while you are in the middle of performing another option or command – block handling, etc, becomes a dream). Uniquely, PERFECTION SE has fully automatic memory management, grabbing and releasing RAM instantly as your document grows or shrinks – programs without this don't take full advantage of the multi-tasking abilities of the QL! Uniquely, PERFECTION SE leaves you in the driving seat, not juggling things around 'underfoot' while you are typing. Uniquely, PERFECTION SE allows up to nine different documents to be handled simultaneously from one copy of the program – each with totally independent margin, tab, justification, control panel, etc, settings. Uniquely, each document can itself have up to six environment settings, each settable or recallable instantly with a single keypress combination. Each document can have any number (up to 500,000 on GOLD CARD) of candidate blocks! Each document can have two independent windows (of any depth, of any (but same) width across) 'on to' it, even with overlapping text – that allows you to edit in one place while viewing another, to compare 'before editing' with 'after editing' (you can arrange to have one window remain 'frozen' in time), etc. Uniquely, we realise how much faster it is to type in something like CTRL/SHIFT/F5 than (say) F3 F3 R – both involve three keys, but as the former doesn't require the keys to be pressed in just one specific order, or to be released in any order at all (together will do), it is in practice twice as fast as the latter, where no key may be pressed until its predecessor is released. PERFECTION SE takes advantage of all this – it is the little things that count! Uniquely, by providing eight user-definable strips, PERFECTION SE allows you to cope with printers of the future, not just the printers that now exist – you can attach the strips to any printer features. Uniquely, PERFECTION SE's status lines give full information on all relevant global settings. And the manual has an index. Also, it has all the important bits at the front.

WYSIWYG?

By the latest definition of this term, neither is PERFECTION SE fully WYSIWYG, nor are other QL word processors. WYSIWYG means what you see on screen is exactly what you get on paper. Exactly – down to every wiggle in every character in every font.

To get true WYSIWYG, use PERFECTION SE's fully automatic link (supplied as part of PERFECTION SE) to PROFESSIONAL PUBLISHER, where you will get 100% WYSIWYG. 100%? Yes, 100%. With this combination, adjust the horizontal and vertical magnification on your monitor (ie calibrate it once and for all so screen circles correspond to same-diameter printed circles – poor monitors may distort a little bit at the edges). Now you can place your printed output from PERFECTION via PUBLISHER over your monitor screen, and get a match that is more perfect than is your eyesight. Now that is WYSIWYG.

SUPERB PRINT QUALITY & FLEXIBILITY

Uniquely, using the aforementioned automatic link, you can output PERFECTION SE documents using over a thousand fonts (a huge variety of styles and sizes, supplied on the PUBLISHER and TOOLBOX disks) on virtually any printer – from the humblest Epson RX80, Brother M1009 or Star LC10 (which are all single font machines when used with most word processors) to top-end lasers. *You are not limited to the fonts built into the printer!!* All PERFECTION SE **bold**/underlined/ *italics*/super/sub, etc, settings are preserved. Proportional spacing and micro-justification are automatic, even when you mix fonts of differing widths *and heights* (even on the same line), vary line spacings, etc. Uniquely, you are not trapped with one type of micro-justification (ie adding all the space between words, and using the predefined widths of characters as their separation) – with our system, you can vary (in 5% steps) the proportion of micro-spaces added between words to that added between characters (the latter in proportion to their *individual* widths). Settings around 65%-35% – not the 100%-0% forced upon you by some other word processors – seem to give the most pleasing results. Uniquely, you are not limited to mere rectangular columns plus headers/footers – that's all the rest can do – you can output in any sequence to any number of frames (text flowing from one to the next), each of any shape – irregular polygons of up to 66 sides, circles, multi-column or part-column boxes (hundreds of types of borders, thousands of textures), doughnuts, wrap-around shapes, even re-entrant ones ('join-the-dots' type borders, even with intersecting edges) – all with micro-justification and proportional spacing! Look at the example on this page. Of course, if super fancy output or special effects are not of the essence, PERFECTION SE's direct printer output is more than capable of meeting your needs.

THE FASTEST

For benchmarking, we've used an unimpeachable file – not one created specially – a public domain version of the first book of The King James Bible, all fifty chapters of the book of Genesis. This came to **one hundred and forty pages**, well over **forty two thousand words** excluding headers and footers, well over **two hundred and twelve thousand characters** excluding justification ones and **one thousand five hundred and thirty three indexed verses**!! We didn't use a smaller file (as used to benchmark other programs) as PERFECTION SE's timings for most operations then become impossible to stopwatch (too fast!). The hardware used for all timings was GOLD CARD: speeds would be **further improved by over three times** using the SUPER GC. Of course, LIGHTNING SE was used. File operations were to ramdisk: normal slave blocks would give identical times. All settings on **everything** were for maximum speed, except where indicated to the contrary – we have the sense **not** to force full speed upon you in operations like scrolling and global Search & Replace. PERFECTION SE's speed for these is switchable (at run-time and when configuring), as too great a speed may cause overshoot (with scrolling) or fatal alteration (if there is human error inputting the target or replace strings). Here are the benchmarks for this huge file:

Load 140 pages: 0.6 seconds (yes 0.6, not 6!) ☆ Import 140 pages: 0.6 seconds (yes 0.6, not 6!) ☆ Save 140 pages: 0.5 seconds (yes 0.5, not 5!) ☆ Export 140 pages: 0.5 seconds (yes 0.5, not 5!) ☆ Case-sensitive search from top for word at bottom: 0.4 seconds (yes 0.4, not 4!) ☆ The same, but case case-insensitive: 0.5 seconds (yes 0.5, not 5!) ☆ Case-sensitive search backwards from bottom for word at top: 0.4 seconds (yes 0.4, not 4!) ☆ The same, but case-insensitive: 0.5 seconds (yes 0.5, not 5!) ☆ Automatic Search & Replace, in Fast (No Query) mode, of last 600 occurrences: 7.4 seconds (same length replace string); 7.7 seconds (shorter replace string); 10.5 seconds (longer replace string – longer time as we deliberately chose a high *density* of replaces to handicap PERFECTION SE into auto-managing memory – without causing any heap fragmentation, but still with only a 0.005 second overhead per replace!). ☆ Automatic Search & Replace in Slow ('Querying') mode: arbitrarily slow, typically 30 times slower – because we deliberately allow for human response time (in case you want to abort) before proceeding from one replace to the next – booby prize to anyone for benchmarking us on this setting!! ☆ Scrolling 100 lines of text, up or down, by full-width screen page: 1.5 seconds ☆ Scrolling 100 lines of text on full-width screen, line by line, in slow (full) mode: 5.7 seconds (down)/5.8 seconds (up) ☆ As above, but in medium speed mode: 4 seconds ☆ The same, but in fast mode and default settings: 13.5 seconds to scroll through the whole massive document, averaging 0.23 seconds per 100 pages (!) – and this could be made up to ten times faster by reconfiguring PERFECTION SE ☆ Reformatting paragraphs, changing margins, justification, etc, of existing text: c5 times faster than predecessor ☆ Inserting (or undoing) emphasised, underlined, italics, superscript, subscript, 8 strips, 6 environment settings: Instant (i.e. immeasurable) ☆ Navigation to line or page or to top or bottom or to 8 markers or to highlights/blocks: Instant ☆ Setting new margins, justification, etc: Instant ☆ Deleting block of 100 pages: 0.3 (yes, 0.3 not 3!) seconds ☆ Copying/moving block of 100 pages (not just 10!), downwards or upwards: 3.4 seconds (yes, including all the time for automatic memory management and anti-fragmentation – other programs are light-years behind) ☆ Spellcheck as you type: Ten times faster than anyone can possibly type ☆ Spellcheck all 140 pages in the document using the 350,000 word Mega Dictionary: 3.9 seconds (20 'errors' – like 'pluckt'!) ☆ And using our tiny dictionary (well, tiny by our standards – large by comparison with most others): 5.1 seconds (566 'errors') ☆ Time taken to create user dictionary from the results of the second spellcheck (566 errors): 0.8 seconds to extract all 'errors' from document and clean document; 1.9 seconds to create a full user dictionary therefrom and also a sorted, duplicate-free wordlist file (for browsing) ☆ Spellcheck file (ASCII or native): Even faster. ☆ Print first 10 pages to file: 3.5 seconds. ☆ Change every occurrence of God to ***God*** in bold underlined italics, strip 2 – 9.5 seconds!

TECHNICAL NOTES. Reformatting is the amendment of a section of previously-entered text to conform to margin, indentation, justification and pagination settings after the user returns to it and makes alterations, either by hand (by over-typing, deleting, adding or otherwise changing) or using search and replace, merge etc. PERFECTION SE lets the user pre-configure, or tune at run-time, the desired reformatting behaviour. The options are to either select Never (most suitable for technical users, and what all previous PERFECTIONs did: you had to initiate the reformat of the re-edited para), Instant (= 0.1 seconds, giving in-situ real-time automatic reformatting as-you-type: common in word processors, and irritating to the eye) or User-delay, the most flexible setting (giving slightly delayed auto-updating of lower text). On User-delay the user is free to set any delay from 0.2 seconds to 99.9 seconds in 0.1 second steps. About 1.5 seconds is best for sedate typists and 0.3 seconds for speed demons. This means that you are not hassled by continuing screen changes on lines below the one you are editing and concentrating upon, or shufflings around on the current line caused by right or centre justification. When you pause in your typing for longer than the set delay, PERFECTION SE automatically tidies up, without you having to do anything. On User-delay, if you navigate or progress off the line, or invoke any menu or command (including Save, Print etc.), an auto-reformat occurs instantly. This means that you are never left with the document in the wrong state. With these options, you have the best of all possible worlds.

Also, SHIFT/CAPS now obeys the indent margin (which matters if the cursor is on the first line of a para) and leaves the cursor position unaltered within the text. Other reformatting commands are unaltered, so you can still step through paras reformatting manually as you go, if you wish. The maximum number of lines, characters, words, lines, pages etc have all been increased (effectively to infinity: e.g., the new limit on characters is 30 million-million, this up from 2 million, restrictive in Super Gold Card / QXL days!). Also, the new version (starting with v5.13) is even faster, and its handling of complex search/replaces (say, involving end of line codes) has been optimised. PERFECTION SE really is superb!

DIGITAL PRECISION PRICES

DIGITAL PRECISION PRICES	£
3-D PRECISION CAD SYSTEM	49.95 d
ADVENTURE CREATION TOOL SPECIAL EDITION	49.95 r
ARCADIA GAME	9.95 s
ARCHIVE DEVELOPMENT SYS + RUN-TIME MODULE	29.95 a
ARCHIVE TUTORIAL	19.95 a
BETTER BASIC EXPERT SYSTEM	24.95 a
BLOCKLANDS GAME	9.95 s
CASH TRADER v3.3 + ANALYSER	99.95 c
COMPARE FILE UTILITY	19.95 a
COPY UTILITY FOR BACKING UP	9.95 a
CPORT BASIC TO C AUTO-TRANSLATOR WITH C68 COMPILER	99.95 c
CPORT BASIC TO C AUTO-TRANSLATOR	89.95 c
DATABASE ANALYSER FOR ARCHIVE	19.95 a
DAT-APPOINT APPOINTMENT DATABASE SYSTEM	19.95 d
DESKJET/LASERJET DRIVER FOR PRO PUBLISHER	19.95 c
DESKTOP PUBLISHER SPECIAL EDITION	39.95 c
DESKTOP PUBLISHER	24.95 d
DIGITAL C COMPILER	29.95 a
DIGITAL C SPECIAL EDITION	49.95 a
DISKTOOL WITH QUICKDISK	19.95 b
DROIDZONE GAME	9.95 a
DR-DOS v7 STATE OF THE ART PC OPERATING SYSTEM	89.95 r
EDITOR SPECIAL EDITION	49.95 d
EDITOR	29.95 a
EYE-Q GRAPHICS SYSTEM	39.95 a
FONT ENLARGER	19.95 r
GAMES COMPENDIUM (ALL FIVE GAMES)	29.95 a
GRAFIX 24-PIN UNIVERSAL PRINTER DRIVER	14.95 r
HARDBACK + FINDER HARD DISK MANAGER	49.95 c
IDIS INTELLIGENT DISASSEMBLER	24.95 a
IDIS SPECIAL EDITION	39.95 d
LIGHTNING ACCELERATOR	24.95 a
LIGHTNING SPECIAL EDITION ACCELERATOR	49.95 h
LIGHTNING GOLD SPECIAL EDITION	39.95 a
MAILMERGE DATABASE SYSTEM	19.95 a
MEDIA MANAGER SPECIAL EDITION	49.95 d
MEDIA MANAGER	24.95 a
MEGA DICTIONARY	29.95 g
MICROBRIDGE	39.95 d
NAMES AND ADDRESSES DATABASE SYSTEM	19.95 d
PAYROLL SYSTEM	49.95 c
PC CONQUEROR GOLD SPECIAL EDITION WITH DR-DOS v7	179.95 g
PC CONQUEROR GOLD SPECIAL EDITION	99.95 g
PC CONQUEROR WITH DR-DOS v7	139.95 r
PC CONQUEROR	59.95 r
PEDIT PRINTER DRIVER FOR XCHANGE	19.95 a
PERFECTION PLUS SPECIAL EDITION WITH SPELLCHECKER	139.95 c
PERFECTION PLUS WITH SPELLCHECKER	99.95 r
PERFECTION SPECIAL EDITION WORD PROCESSOR v5	99.95 c
PERFECTION SPECIAL EDITION DEMO VERSION + SAMPLE DATA	19.95 c
PERFECTION WORD PROCESSOR v3	59.95 d
PERFECT POINTER TOOLS	29.95 a
PROFESSIONAL ASTROLOGER WITH ASTRONOMER	69.95 a
PROFESSIONAL ASTROLOGER	59.95 a
PROFESSIONAL ASTRONOMER	29.95 s
PROFESSIONAL PUBLISHER TOOLBOXES (BOTH PARTS)	49.95 r
PROFESSIONAL PUBLISHER TOOLBOX PART ONE	29.95 r
PROFESSIONAL PUBLISHER TOOLBOX PART TWO	29.95 r
PROFESSIONAL PUBLISHER	89.95 c
QFLICK CARD INDEX SYSTEM	29.95 a
QKICK FRONT END SYSTEM	24.95 a
QMATHS MATHEMATICAL SYSTEM PART ONE	69.95 c
QMATHS MATHEMATICAL SYSTEM PART TWO	59.95 c
QMATHS MATHEMATICAL SYSTEM (BOTH PARTS)	99.95 c
QMON MACHINE CODE MONITOR	39.95 a
QUICKLASER PUBLISHER DRIVER FOR LASERJET/DESKJET PRINTERS	19.95 c
RECOVER ARCHIVE UTILITY	19.95 a
REVERSI GAME	9.95 s
SEDIT + SCREENPRINT ARCHIVE UTILITIES	19.95 a
SOLUTION PC EMULATOR WITH DR-DOS v7	109.95 r
SOLUTION PC EMULATOR	29.95 r
SPELLCHECKER WITH DICTIONARIES	49.95 d
SUCCESS CP/M EMULATOR	49.95 b
SUPERBASIC MONITOR	24.95 a
SUPERFORTH COMPILER WITH REVERSI	39.95 a
SUPER ASTROLOGER	24.95 s
SUPER BACKGAMMON GAME	9.95 s
SUPER SPRITE GENERATOR	29.95 a
TOOLKIT III WITH ROM	49.95 h
TOOLKIT III	29.95 a
TRANSFER UTILITY SPECIAL EDITION	29.95 b
XREF SUPERBASIC PROGRAM ANALYSER	29.95 a
TRANSFER UTILITY	9.95 b
TURBO BASIC COMPILER + TOOLKIT	79.95 a
ULTRAPRINT SCREEN DUMP UTILITY	19.95 a
XREF SUPERBASIC PROGRAM ANALYSER	29.95 a

KEY a=Available either on cartridge or disk; b=Disk only; c=Minimum 512K exp: disk only; d=Minimum 256K exp: either cartridge or disk; r=Minimum 256K exp: disk only; s=Cartridge only; g=Minimum 1.5Mb RAM: disk only disk; h=ROM + (cartridge or disk)

AND NOW - BEST-EVER SPECIAL DEALS...

5% off 2 programs
10% off 3 programs
20% off 4 programs
30% off 5 programs
40% off 6+ programs!

SUPER GOLD CARD

This amazing product is the way forward for the QL. Like Gold Card before it, the brand-new **Super Gold Card** is a plug-in disk interface and RAM expansion that works on all QL versions. Incredibly, it is *over three times* speedier than Gold Card, with *over twice* the RAM and with many enhancements. It complements LIGHTNING SPECIAL EDITION like nothing else, squeezes the best out of TURBO (which was designed with 32-bit CPUs in mind) and really accelerates PC CONQUEROR. **Super Gold Card** is actually as fast, or slightly faster, than the much vaunted QXL: also, it is **100%** QL-compatible **now**, and no PC is needed. The table below really says it all:

System → *↓ Features*	*Bare* *QL*	*TRUMP* *CARD I*	*GOLD* *CARD*	**SUPER** **GOLD CARD**
Relative Speed	1x	1.8x	7x	**25x !**
Motorola CPU	68008	68008	68000	**M68020**
Clock Frequency	7.5MHz	7.5MHz	16MHz	**24MHz**
Bus width	8 bit	8 bit	16 bit	**32 bit**
RAM fitted	128Kb	896Kb	1,920Kb	**3,968Kb**
RAM access speed	Slow	OK	Fast	**Twice as fast**
PCB population	V.high	High	V.low	**V.low**
Physical dimensions	Monolith	Full-size	Half-size	**Half-size**
Lock-up frequency	Ouch!	Occasional	V.rare	**Won't**
Battery Backup Clock	No	No	Yes	**Yes**
Clock Protection level	N/A	N/A	Modest	**High**
Toolkit II + Manual	No	Yes (early vns)	Yes	**Enlarged**
Sub-directory support	No	No	Yes	**Yes**
Parallel/Centronics port	No	No	No	**Yes**
Spooler/Screendump/Ramdisks	No	Yes	Yes	**Yes**
Speedup switch (Screen#2)	No	No	No	**Yes**
Future hi-res graphics	No	No	No	**Planned**
Disk drives supported	N/A	SD/DD	SD/DD/HD/ED	**SD/DD/HD/ED**
Max no: of disk drives	0	2	3	**4**
Max sectors/disk	N/A	1,440	6,400	**6,400**
Max disk transfer rate	N/A	30Kb/sec	120Kb/sec	**>120Kb/sec**
Peripheral card tolerance	OK	No	No	**OK**
SCSI-2 compatibility to-be	No	?	No	**Yes**
Miracle/DP Warranty	No	No	2 years	**2 years**
DIY/Kit incorporability	Yes	No	No	**Yes**
Overall Rating by DP	2%	10%	30%	**110%**

And to the Very Best news: from DP, SUPER GOLD comes SUPER CHEAP!
SUPER GOLD CARD, plus a no-limit extra 20% SOFTWARE DP DISCOUNT VOUCHER, plus a FREE mystery DP program, plus a FREE Dust Cover, will cost you a mere £375 ✓✓✓✓
Less £125 if part-exchanging your standard 2Mb Gold Card
Add £125 for ED 6400-sector Disk Drive (PSU, cased, cables).

OTHER HARDWARE EXCHANGED AND SOLD BY ARRANGEMENT. PLEASE ORDER NOW: WE EXPECT VERY HIGH DEMAND FOR SUPER GOLD CARDS, AND WE DON'T WISH TO DISAPPOINT. INTERNATIONAL RAM PRICES ARE UNSTABLE AND PRICE HIKES MAY BE INEVITABLE. CONSEQUENTLY, THE ABOVE COMBINATION OFFER COULD BE WITHDRAWN WITHOUT NOTICE...

The software discount must be taken at the same time as the main order, and CAN be combined with the SPECIAL DEALS discounts. For example, if you chose six DP programs of total list price £100, you would only have to pay £100 -40% -20% = £48 for them! And, of course, you would also get the two gifts absolutely free, and a 4Mb SUPER GOLD CARD too!

TERMS & CONDITIONS ** We accept payment by VISA, Mastercard, Access, Eurocard, Eurocheque, £ cheque drawn on a UK branch of any bank or building society, postal order, cash, travellers cheques or direct funds transfer to our bank account #60327808 at Barclays Bank plc, South Chingford Branch (sort code 20-53-00), NE London Business Centre, PO Box 2403, London N18 2BY. If you must use another currency, add 7.5%. If you must use another form of cheque, add £15. * Upgrades cost only the difference in current price plus £10; return the original disk and manual. But do NOT return the manual if the upgrade is to a "SPECIAL EDITION" of your original program: you will be sent useful supplementary documentation. Upgrades from LIGHTNING are the exceptions to the last rule! * Only DP programs and upgrades (i.e. not DR-DOS) count towards discounts. * Prices are all-inclusive for UK mail-order. Rest of Europe, add 5%. Elsewhere, add 10% (Air Mail). * We aim for 24 hour turnaround. * DP & Digital Precision are trading names of Digital Precision Ltd, Company No.1833989, registered in England & Wales. * Orders using your own paper are also welcome!*

✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂

Post to: *DIGITAL PRECISION*, 222 The Avenue, London E4 9SE, UK
or phone on 081-527 5493 from UK, +44 81-527 5493 rest of the world

Name:________________ Address:________________

____

Products ordered: ________________________

____

on mdv/3.5"disk/5.25"disk* (no: of sectors on disk-format = 360/720/1440/2880/6400*; if 5.25"/720, specify 40/80* track) *=*delete as needed.*

Signature: ________________ Amount: £________

Card No: ________________ ________ Expiry date:________

QL Contents

VOL. 3 ISSUE 1

Editor
Helen Armstrong

Publisher
Mark Kasprowicz

Advertising Manager
Jim Peskett

Creative Director
John Stanley

Graphic Artist
Stevie Billington

Magazine Services
Linda Miller, Frances Maxwell, Pauline Wakeling

Sinclair QL World,
Published by Arcwind Ltd.
The Blue Barn,
Tew Lane, Wootton,
Woodstock,
Oxon. OX7 1HA
Tel: 0993 811181
Fax: 0993 811481
ISSN 026806X

If you have any comments or difficulties please write to the editor and we will do our best to deal with your problem in the magazine, though we cannot guarantee individual replies.
Back issues are available from the publisher price £2.50 UK, £2.99 Europe. Overseas rates on request.

Subscriptions from: Arcwind
The Blue Barn, Tew Lane,
Wootton, Woodstock, Oxon.
OX7 1HA
UK: £23.40
Europe: £32.90
Rest of World: £40.90

Reprographic Services:
Eclipse, Brook Street,
Watlington, Oxon. OX9 5JH.
Distributed by: Seymour Press Ltd., Windsor House, 1270 London Road, Norbury, London, SW16 4DH

Coming Soon

World Map ... More on Xchange, Perfection, Quill, Abacus ... QL front end.

QL Scene

PAGE DESIGNER - THE NEXT GENERATION!

DJC now has a completely re-written **Desktop Publisher, Page Designer 3**, THe Next Generation, ready for use.

PD3, the new incarnation of old favouite Page Designer two, has been entirely rewritten ("well, almost") by author Barry Ansell for the QL in the 1990s, operating under todays' extended QL environment to use pointer (mouse) control or keyboard control. "QL Desktop Publishing Made Simple!" is the program's motto. Even so, it is a serious tool, and needs minimum 512K and SuperToolkit II to run. It is also Gold Card and Minerva compatible.

The four-disk package includes, with DP's permission, Digital Precision's High Definition Font format, and new and more efficient built-in printer routines.

The A4 instruction manual deals with the program menu by menu, and also has detailed information about page sizes, file checklists for all four disks, a separate section on the Font Editor, and printed pages of clipart for guidance.

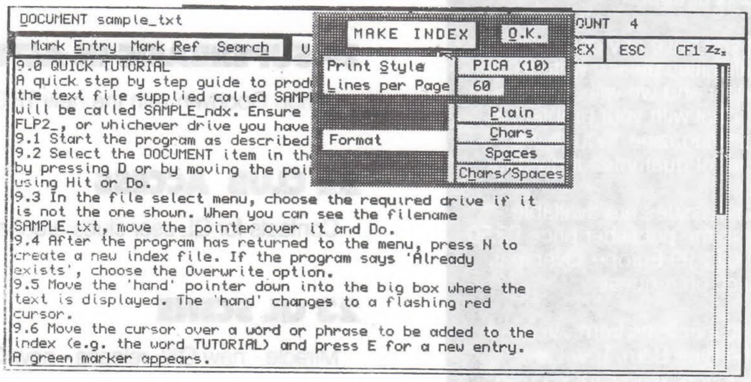
PD3 costs £40 (Post free in UK, postage £1 per program (max. £3) overseas), or an update kit for £25. Various clipart packs are available for £10 or £12. All orders and requests for information to **Dilwyn Jones Software, tel. 0248 354023**.

QL World would like a review of PD3 from an experienced PD2 user or desktop publishing user - we can supply software - and in due course we would like

Complete Indexer from DJC

New from DJC is **Qindex**, a program to assist in producing sorted indexes for a book or file. Unlike database-type indexing programs, where the entries are typed into a list, you can load into this program a plain text file or Quill _doc, and flag words and phrases directly to be inserted into an index. The resulting index can be built up from several text files (such as chapters of a book or manual), not just one.

There is a Search facility to find all occurences of an item to be indexed. Entries with similar references can be automatically combined. There is an on-screen facility to count the number of entries.



Once finished, the index can be **exported to a text editor** (if desired) for restyling etc. prior to printing. The program has a choice of text pitches from 10 cpi to 20 cpi, allowing layouts of different numbers of columns. Other layout choices include: entries beginning with different characters can be separated with a blank line, or a line containing the single character as a header, for instance. Users can devise their own layouts, or move the index to another editor for layout.

Indexer is Gold Card and Minerva compatible, and works in the Pointer Environment (supplied), so that a mouse can be used if desired. It can also be driven entirely from the keyboard.

The program is on disk only (3.5 in or 5.25 in) and needs an expanded memory and Toolkit II (normally built into the disk interface). A complete printed manual is included.

QL World would like a review from an experienced indexer in due course. Indexer costs £20. Cheques etc. should be made payable to **Dilwyn Jones Computing. Tel. 0248 354023**. For address etc. see Dilwyn's ad. on the back cover of this issue of QL World.

Fairs Now!

Don't forget the **Quanta/ SQLUG workshop** at Napier University, Colington Road, Edinburgh on February 12th - contact **Alan Pemberton on 031 660 1826** (early evenings) for information.

And the **International QL Meeting** at the University of Bielefeld, Germany (near Munster) on February 19th - contact Franz Herrmann, E-mail: Franz_Herr- mann @ bn.maus.de D-53545 Ockenfels for information.

We have some last minute infomration about exhibitors and accommodation today: Miracle, TF Services, Progs, Qubbesoft, EEC, Grange Technology, Jurgen Falkenburg, Albin Hessler, NASA (Norway), Ergon Development, QItaly, Oliver Fink (QSpread) and Jochen Merz are among those expected.

For more information reach Andreas Rudlof on ++ (010 from UK) 49 2242 85515, Franz Herrmann on ++49 228 231480 or the Bielefeld Tourist Information office on ++49 521 178844. Andreas has a faxable list of information and hotels for travellers.

QL Scene

QL STOCKS SOLDIERING ON

Bill Richardson only has about 20 of his famous New QLs left! The **unused QLs**, complete with original packing and manual, are the last of the shipment bought by Bill from their warehouse several years ago, where they had apparently been laying neglected since the "boom days" of QLdom.

Once this elite band have homes, it looks as though the last of the "first owner" QLs will be gone for ever. The New QLs are still £120 for a JS package and £100 for the JM, and the price of a New QL includes a free one-year membership to Quanta.

Bill accepts **used QLs** in any condition in part exchange, giving a £30 discount on the New QL prices. "We collect these for spares", Bill explained. Richardson's "Backup QLs" - just the QL and its PSU, with no paperwork or software, are on sale for £80 for JS and £65 the JM rom versions. The part exchange allowance on the purchase of Backup QLs is £15.

As for **replacement keyboard membranes**, "QL membranes are OK, but not the Spectrums." It's too expensive now to order a whole batch of Spectrum membranes with sales very low. However, QL membranes and instructions for fitting them cost £9 from Richardson's. Other spares getting scarcer but still around are 8301 and 8302 chips for the QL, and ULAs for the microdrive units.

Another threatened species is the **microdrive cartridge.** Manufacture of cassettes ground gradually to a halt some four years ago as documented in QL World, but steady sleuthing of unused stocks and second-hand cartridges by people like Bill has kept supplies coming. Bill reckons he has 3-400 left, and a sales turnover of about 200 a month, so if you want more microdrives, speak to him soon. Sample prices are 8 walletted second user cartridges, £15; filing box with 20 new cartridges, £45.

Hardware designed by people currently in the QL market is selling steadily - Jugen Falkenburg's I/O card at £235, and keyboard interfaces for PC-style keyboards, a popular QL mod, at around £75, depending on the kit.

Phone Bill on 0753 888866 or send a medium SAE to W N Richardson (EEC), 18-21 Misbourne House, Chiltern Hill, Chalfont St. Peter, SL9 9UE.

Enigma Solution

Here is the solution to the **coded message** from the Engima program printed in QL World II.9 (September 1993).

Decoding the first six letters using the Code Indicator Setting CAT will reveal the Main Code Key GNU from which, after re-setting, the message can be decoded as:

"If you are looking for a magazine which will stretch your horizons and expand your knowledge of this versatile little machine, the Sinclair QL World is for you. Sign up for a year's subscription now and take advantage of the latest offer. You'll never regret it."

(George wrote that all by himself! He's obviously due for a very successful second career as a diplomat!)

Of course, there will not be any spaces, punctuation or capital letters. This is a mesage of 211 characters, which together with the sic characters of the Main Code Key constitutes the total message length of 217 characters. Remember that you can easily recover from an incorrect letter in the code, but never from a character omitted! If anyone wants more information, just write to us.

Machine Code Listings Errata

The followers of Alan Bridewell's **Machine Code For Beginners** were on the phone to use as soon as their QL World II.12 hit the mat. Anyone who wants a replacement copy of the pale listings on pages 27 and 28 (and also on page 21 in DIY Toolkit), or any of the other listings in those articles, write to me with your address and which listings you want, and we will send you a fresh copy. Allow a little time, as we have to get the originals back from the printers (or from Alan.)

Some Machine Code beginners may want a reprint of Listing 1 from **part 6**, issue II.9, in which some lines were omitted due to an obscure

All Formats Diary

In 1994, the All Formats Computer Fairs will be held **monthly at three northern venues**, and only occasionally in other places.

The monthly fairs will take place at the National Motocycle Museum, junction 6 of the M42, opposite the National Exhibition Centre entrance (a very central location for many parts of the country), Haydock Park Racecourse, J23 of the M6, between Manchester and Liverpool, and the Washington Leisure Centres at the top of the A1(M).

A smaller number of fairs will be held in Bristol, Glasgow, the South East, Belfast and Cardiff. It seems that the All Formats Fairs have dropped out of London altogether.

In the next two months the following Fairs are lined up: **Jan 23** West Midlands Motorcycle Museum; **Jan 29** Essex Brentwood Centre; **Jan 30** Surbiton Southborough School; **Feb 12** Haydock Park; **Feb 13** National Motorcycle Museum; **Feb 26** Northumbria Centre, Washington; **Feb 27** Glasgow Woodside Hall.

The events are open from 10am to 4pm, entrance adult £4 (£3 with voucher), £2 child; £2 all comers after 2pm. A supply of "£1 off" vouchers can be obtained by sending an SAE to **All Formats Fairs, Maple LEft, Stretton-on-Fosse, Moreton-in-Marsh, Gloucestershire GL56 9QX.** Stands at the fairs cost £60 tel. 0608 662212.

OPEN CHANNEL

Open Channel is where you have the opportunity to voice your opinions in Sinclair QL World. Whether you want to ask for help with a technical problem, provide somebody with an answer, or just sound off about something which bothers you, write to: Open Channel, QL World, The Blue Barn, Tew Lane, Wootton, Woodstock OX7 1HA.

Two Threes

Following the review of Digital Precision's **QMaths 2** mathematics collection, I would like to point out that there are, in fact, two programs called **"3D Terrain"** knocking about on the market at the moment!

The other program was previously distributed by CGH Services (RIP) and is now handled by Dilwyn Jones Computing. My reasons for pointing this out are (a) pride and (b) blatant plugging of the product, since I co-wrote this version!

The program included in the DP package is of American origin, and regular readers may recall that Bryan Davies reviewed both programs about a year ago, a couple of months apart. I am not rubbishing one in favour of the other! The two are very different and only share their name by chance. In fact, I would urge interested readers to invest in both, since they fulfil different roles and are both low in price.

If I may be allowed to briefly highlight some of the differences: the US program is a complex shape generator with a very good graphical capabilities (viewpoint, scale, colour, etc.), whereas the GB program is a spreadsheet viewer, taking any exported Abacus spreadsheet data and supplying the results as a 3D surface, therefore allowing the user to input data.

The US program produces a screen image by calculating every pixel position, rather like a fractal viewer. The GB program is only concerned with calculating the end points of straight lines, which it combines with other similar lines to form a surface grid created from "diamond"-shaped elements. *(Note: this is a bit like the difference between colouring and texturing techniques in virtual reality, which trade off absolute detail against speed, depending on the application.)*

In anyone is interested in this history of the GB program, I originally wrote my version because I got dizzy looking at OS maps of the very hilly areas near home where I went mountain-biking. Therefore, the name "3-D Terrain" seemed appropriate. In 1989, after successfully modifying Mike Lloyd's maths engine from his 3D graph series of the previous year, I asked Richard Alexander of CGH Services if he thought much of the idea, which he did. The necessary copyright information was sorted out with Mike, but progress from the original to something presentable was slow (my fault!). Rich Mellor was given the task of compiling the program using Turbo, but in the end he modified the original code and improved the "feel" of the program in many areas.

From the outset, it was a hobby type of program, which is why it retains some of its amateur charm! Once the Basic loader program is set up to cope with the size of the latest version of DP's Lightning, and the Psion GPRINT_prt file is copied across to the host medium, all is fine. The lack of demo files (there are only five) is again my fault, since it was intended to fit on an unexpanded, diskless machine. The Basic development versions ran on a 128K QL, but the compiled version needs extra memory, although I think it may still squeeze onto a microdrive.

The user can very easily enter data via Abacus, or anything that will Export data to Easel in the same way as Abacus. Endless demo files are relatively pointless, since 3D-Terrain (GB) is a spreadsheet rather than a complex equation displayer, and the idea is that the users "get grubby" and stick their own numbers into the program. Of course, the data doesn't have to describe a landscape, it could be rainfall, pressure, temperature, basically any variable expressed over an area. The biggest demo file displays an actual landscape, but, as mentioned in the manual, the program can handle four times as many data points than this image contains, limited by screen resolution, which ultimately affects all display programs. Gimmee a QXL with SVGA capabilities, please!

Sorry this letter has grown into a book! I hope it clear up any confusion, and people don't ring Dilwyn thinking it's the same program as is in the DP collection.

Ian Thompson
Ripon

Wrong Definition!

Soon after I wrote my last letter to you I discovered an article *[not in QL World or Quanta]* on **Quill in Xchange** which explained how the glossary worked. In the _doc files on the disk it only says: "These settings are automatically saved when you quit Quill". Wrong. They aren't. The article adds: "By the QUIT command pressing 'A' (New - A=Abandon, and will only appear within the command box when defining a glossary file." This is also wrong. If you type anything into Quill and try to quit before you save it, this option always crops up.

I experimented with this command and found it did save a glossary, but only once and then that was it. I have not been able to save a glossary since, and I cannot edit the original which I did in a hurry, and only as a trial. What is happening here?

I must confess that I have problems with computer manuals and often have to read them several times before understanding what I am really supposed to do, so I often make mistakes. I spent ages trying to work out why the date on the **Minerva** clock was always slow when I reset it, until I noticed that the time has to be reset in the Mini-config program. This and your request for reviewers for certain programs recently leads me to suggest that you should run two review teams. One is "the experts", who can read the manuals, extract the relevant information and run the program first, and the other, a collec-

tion of people who, like me (there must be plenty of others) get totally lost in the instructions and technical language.

I can, eventually, get things happening. Having rewritten the Boot file about twenty times, I now have QPAC2 running beautifully and my problems with SERmous only occurred because I did as the adverts suggested and bought a cheap mouse. (Jochen Merz supplied me with a better one, and now that is wonderful as well.) So maybe reviews by people like me might be a good idea to show the obvious mistakes that the experts would immediately notice and cure without a mention.

Roy Wood
Hamburg
Germany

We try to make sure our reviewers refer to the manuals and discuss them, so that if they are hard to understand, people will be warned. We strive to give an idea how good the software is overall, as everyone has some difficulties with new programs, often for reasons of compatibility with printers, or personal misunderstandings which it would not be fair to highlight in a general review. However, we want to know if anyone has persistent difficulties. Always get in touch with the manufacturer first if you have a problem. Most QL suppliers give good support (as every supplier should).

Your method of writing a Boot program - doing it 20 times over until it works well - would strike many of us as absolutely correct! There is a lot of "if at first you don't succeed" in programming.

TRA

I was hoping to see the definition of the Basic command TRA in the useful series the New User Guide. True, not all roms support this, but JS (presumably quite widespread) does. I find the command useful for printing such characters as "up arrow" and "right arrow" on my Epson printer. Indeed, there is oblique reference to TRA in the section CREATE_PRINTER_DRIVER in Digital precision's manual for Perfection. So for those who are interested (and there must, or should in my opinion, be many users of Perfection) here is my definition of TRA unravelled from its machine code.

TRA address1 [address2]
(JS and above)
Translation Command

This command controls two tables, Serial Translation and Message. The first translates output to SER1 and SER2, the second provides the error messages.

The default table translates each byte to itself (ie it does nothing). This can be changed by specifying in address1 the position of a user table. This should contain the 7 bytes 74,251,0,6,1,6,0 followed by 255 bytes being the translation of bytes 1 to 255. (Note that byte 0 is never altered.) After this is an extension table. If any byte in the translation table is zero then three bytes are taken from the extension table which is made up of a byte giving the number of translations followed by that number of groups of four bytes, the first of which is the byte to be translated, the remaining three being the three bytes actually sent to SER1 or SER2.

If address1 = 0, no change is made to the current Translation Table.

If address1 = 1, the Translation Table reverts to the default.

Otherwise address1 is taken as the address of the new Table.

A user Message Table can be set up by putting address2 equal to the address of a user table. This table must start with the two bytes 74,251 followed by 29 words giving the offset of each of the 29 messages frost he start of the table. Each message except the last two consists of one word containing the length of the message followed by the message itself. The last two messages have no initial defining word. In fact, only 21 of the messages are true error messages, so the others ought to be copied to end of the user table from the default taxable. The address of the Message Table is held in SYS_VARS + $14A (usually $2814A).

If address2 = 0, the Message Table reverts to the default.

If address2 is omitted, no change is made to the current Table.

Otherwise address2 is taken as the address of the new Table.

George Gwilt
Edinburgh
Scotland

Back Numbers

In Issue 10 I read of requests for back numbers of QL World. I have all copies of QL World, QL User (including the Special Launch Edition) and QLUB News issues 4-8... I am willing to supply photocopies of requested articles. I do not have my own photocopier, but a local business will make copies for contributions to charity of 10p a sheet, and these could be supplied at that prrice, plus price of a large envelope and postage. I have my own index of articles, not perfect but I can usually find things.

On my copy of D-Day the first request is "Is there a ram disc available?" I,m sure there is but I don't know how to find it. Words of one syllable, please.

Arthur Nunn
Ripon
0765 689378

Editor's Notebook

If any of the editorial pages look odd this month, it's quite likely to be my fault, as I have entered most of them, in the nightmare company of Hell's Monitor, which kept fading back into the darkness. whence it came. We examined its insides and threatened it with a major operation, and that seems to have settled it for a while...

This month we have news of an interesting new indexing program from Dilwyn Jones, and it looks as if Miracle, faced with directing their energies in many different difrections, are making determined plans to be in full control of as much of their sfotware, as well as their hardware, as possible for future developments. They also expect to have the first-demo SCSI card ready for Newport in May.

Please note that Qubbesoft's workshop number is 0376 347852, and not the one printed in last month's QL Scene. Sorry, Ron.

TROUBLE SHOOTER

Bryan Davies looks at the habits of computer founts.

Full marks to Hewlett-Packard for service. Alf Kendall has an H-P **DeskJet** printer, which went sick a few months back. He thought the printer was no longer covered by warranty, being more than one year old, but, when he contacted Hewlett-Packard he was asked for the printer serial number and told it was still covered. The printer was duly collected - yes, H-P arranged for it to be picked up - on a Friday. arrived back, working properly and with several parts replaced, the following Friday. This was a couple of weeks before Christmas, when one can usually expect delays. Advice with the printer told him the warranty extends until spring 1994, well past the end of the one-year nominal warranty period. As you might expect, Alf is a very satisfied customer.

The DeskJet is a good printer, and justifiably popular. There can be problems with the ink feed if the machine is not used regularly. As with most inks, DeskJet cartridges can clog, so keep it flowing freely by printing the odd page every few days. If the printer has to be left inactive for weeks, remove the cartridge, and cover the nozzle area; it might also be worth putting the cartridge back into its packet and sealing that with tape as tightly as possible.

Ink cartridges are expensive, but the costs can be kept down with refills. Rather than waiting until nothing at all is coming out of the nozzles, inject a little ink into the cartridge at regular intervals. A refill equivalent to the amount in two new cartridges costs about £8-12 plus VAT.

My experience with the Canon BubbleJet BJ-10EX is that a cartridge left for months without use is too far gone to be worth trying to revive with a refill kit. You can get it printing again, but the quality is not good. Possibly, soaking the cartridge in a suitable solvent for several hours would do the trick if you have more time than money.

Printer Founts

Bryan Orgar is puzzled by founts. Like most micro users, he is attracted to improving the appearance of his printed output, but finds it difficult to understand the terminology and techniques. Printing causes more heartache for most of us than any other facet of home computing, but life "before founts" was a whole lot simpler than it is now.

Printers come with **"built-in" founts**. These appear on the test prints, which can be done without the printer being connected to a computer at all. These founts are comparable with those on the screen of the basic QL, as they are "library sets" of characters, default founts. Over the years, the built-in founts have increased and improved, and even fairly cheap dot matrix printers now have a reasonable set of founts. However, many people are not satisfied with the basic founts, and look around for additional ones.

This is where you get on the slippery slope. There are two main categories of additional fount - those on plug-in fount cards or cartridges, and "soft founts". As the name implies, plug-in cards contain extra founts, which effectively increase the built-in set. These founts act just like the built-in ones. One major drawback of founts on cards is their cost; they usually work out far more expensive than software founts, and I never recommend them, for this reason.

Downloading Founts

Software founts come in several varieties, but the general principle of operation is much the same. The fount characters are stored, or generated, in the computer and "downloaded" to the printer. The manner of downloading is not the same for all types of soft fount. One technique is to send complete fount sets to the printer before printing starts, or as required during printing. This may need plenty of storage in the printer; fount files can take up a lot of space.

Some types of printer have little or no memory available for storing founts. Dot matrix printers and inkjets come with far too little memory for whole sets of characters to be stored. For these, it is necessary to send fount characters more or less one by one, rather than whole libraries at once. There are considerable variations in speed between different techniques. Downloading a whole fount is a time-consuming business but, once it has been done, the founts become effectively built-in and there is no great delay in printing from there on. Sending characters individually means having to send some of them repeatedly, and this is a slow process.

Soft founts can be stored in the computer as library sets, just as the basic founts are stored in a printer, except that the soft founts will be on hard disc whereas the printer ones will be in a rom chip. A big disadvantage of the library fount is that it has to be of a fixed size. For instance, Courier 10-cpi (characters per inch) is one of many fixed sizes of the Courier fount, and can be stored in one fount file. If you want 12-cpi as well, that means another file. Add together the founts you want, and the sizes in each, and you arrive at quite a few files. File size tends to increase with Character size, so that different sizes of the same basic type (eg Times) can range from, say, 10 KB up to 100 KB.

An alternative is to generate each **size on demand**. This is what happens with scalable founts. The basic designs of the characters are stored, as a collection of mathematical definitions, without any size being attached to them. When the request comes for a particular size, the appropriate multiplying factor is applied to produce the desired size. The resultant character is sent to the printer. This is a

very flexible approach, which gives good quality print over a wide range of sizes. The fact that the characters are not stored as dots in a matrix means that enlarging them does not result in a corresponding increase in jagged edges.

Scalable Founts

Scalable founts do not have to be soft founts. They can be "firm founts", stored on firmware (on a chip). Some types of printer - notably inkjet and laser - come with built-in scalable founts. These are invariably preferable to the fixed-size founts which were standard in dot matrix printers for many years (and still are in cheap printers).

Before switching to a laser printer, I had thought that the dot matrix printer had some advantages, but that idea soon disappeared. In almost every way, the dot matrix printer is less flexible. **Perhaps the only advantage is the ability to take continuous stationery.** Even the ribbons can be very expensive, comparable in price to ink cartridges and laser toner. The cost of a colour ribbon for the Star LC24-200, for example, is about £10-11 plus VAT, and I am told that they do not last long. There are some inherent design weaknesses in dot matrix printers which cause trouble. One is the inability to print near to top or bottom of paper, because of the physical spacing between print head and bail bar, and the need to have enough paper to grip to keep it aligned. Another is noise.

Inkjets have their faults, too. They tend not to have emulation modes compatible with your existing printer; this can make them unusable if your application programs do not have your specific printer drivers. You cannot simply use an Epson FX-80 driver and get at least a basic printout, as you can with most dot matrix printers.

In principle, dot matrix printers can print the same soft founts as inkjet and laser printers. Bryan Orgar has a NEC Pinwriter P2200, which is usable with many soft founts. The difficulties arise with individual application programs and their printer drivers - or lack of them - and with individual founts. Bryan also uses WordPerfect, and he should have no great trouble finding a suitable driver for that program.

Create Printer Data

However, he prefers his Perfection SE to WordPerfect. Digital Precision do not have a vast stock of drivers, but they allow you to custom-build your own, using their Create Printer Data routine and inserting the appropriate control codes for your printer. This can be be painful with some printers, such as the H-P lasers, but should be no great job with the P2200. DP will give advice on creating drivers, but don't expect them to spend hours writing one for you at no charge.

The Perfection driver can utilise the Strip function to enable the various built-in founts to be selected. As yet, the QL has not been favoured with masses of soft fount programs. We have had plenty of founts offered, but they generally are unusable by the average person, because they are not automatically made available to the application programs. They can be made to replace the on-screen QL founts, which may create a pleasing visual effect but is no use for printing.

Text87 has a graphics driver which enables the founts supplied with it to be printed exactly as they are displayed on the screen, a rare genuine WYSIWYG (what you see is what you get). There used to be a very good program (CuePrint) which provided masses of well-designed founts for dot matrix printers, but that program is, alas, no longer available. For the moment, there is no doubt that you are better off with an inkjet or laser printer, simply because they have better built-in, and scalable, founts. Make sure your application program supports the printer you choose, though.

When thinking about soft founts, you may need to forget characters as discrete ready-made objects. The tendency now is for text to be printed as graphics: each character is created, and sent to the printer, as a "picture".

The expression "soft founts" was coined to describe library character sets. The main difference between soft founts and built-in printer ones is that soft founts are stored in the computer.

On the other hand, graphics founts, such as those used with the Windows interface, are "soft" in that they are created by software, and are stored on hard disk, but the characters are transferred to the printer as individual images. Some founts have both screen and printer files, but the Windows type now generally use the same files for display and printing.

Graphics Printing

It is easy to be overcome by the wonders of new founts, but do not lose sight of the down-side. Graphical printing is usually much slower than printing with built-in characters - this makes a lot of difference when you take into account that the QL can be speeded up very little when printing graphic images.

Using built-in founts is much faster. Also, text quality sent graphically from the computer to a dot matrix printer can be mediocre on paper, worse than built-in founts. You can trace a path of decline for print quality, from the typewriter-quality of the daisywheel, down to the near-letter-quality on the dot matrix, down to graphic text to the dot matrix.

Laser and inkjet printers do a very good job of printing graphically; much better than even a good dot matrix printer does.

Odds and Ends

The word has been passed to me that my chemical terminology is out of date. Specifically, isopropyl alcohol is now normally referred to "in the trade" as isopropanol. Likewise with ethanol, methanol etc. If you happen to be a drag-racing fan, methanol should have been a familiar term for many years, but literature on disk cleaners doesn't often use the term isopropanol. Now we know.

The liquid itself is an evaporating alchohol cleaner, used in disk drive kits to clean the recording heads. It can be used for many other cleaning jobs, such as on the record heads of microdrives. It appears not to be strong, in the sense it is not liable to damage surfaces it comes into contact with, making it fairly safe to use for cleaning plastics.

A supplier called Furst (?) is said to be offering the SMS2 operating system on rom chip for the Atari, at a price of about £135. There is a dedicated rom socket on some Atari motherboards, and the SMS2 chip fits into that; this approach sounds to be somewhat different from the QL emulators sold up to now. I heard this on the grapevine. Has anyone any more details?

Review QVME

The future now? Wolfgang Lenerz uses the QL emulator for the Atari STe.

INFORMATION
Prices: QVME card: DM 695
Adapter: DM 90
Supplier: Jochen Merz Software, Im stillen Winkel 12, D-4100 DUISBURG 11 Germany

For some time, the **Atari emulators** were the best and only way to get a better, faster and more reliable QL: buy an Atari, install an emulator card (which involved some brandishing of the soldering iron), and the Atari behaves like a QL. As the Atari had a true 68000 (instead of the QL's 68008) it was noticeably faster than the QL, especially in printing to the screen, as it is not hampered by the QL's double access to the screen memory. A Mega ST with 4 MB memory and a hard disk makes a formidable QL!

When Atari brought out the STe (which is faster than the older ST series, being a 16 MHz machine), everybody waited for Jochen Merz to come up with an emulator for that machine - the old emulators would not work with it. That new emulator is finally here. It was very long in coming, and I suppose that Merz lost a few sales to the Gold Card by then (he says he didn't). It is called the QVME card, because it plugs into the "VME bus" of an Atari Mega STe.

On The Card

There is not much to say on the physical description of the QVME card. As you can see from **Figure one**, it is neatly finished, with a VME connector at one end. I confess that I was astonished by the number of chips on it - the board is densely populated, even though it uses high integration. It is a very professional product, there is no evidence of last minute tampering, or wires running where they shouldn't.

Nice as the card may be, were it just another emulator, only adapted to the STe series, it would hardly be important enough to merit a review in **QL World**. But there is more to it.

First of all, installation is very much simplified: The card simply plugs into the STe's VME bus. No soldering is required. The QVME card can only be used in Mega STes, as the 1024 STe does not have a VME bus. Given the country of origin, this apparent restriction of choice is not surprising: In Germany, the majority of Atari machines sold were Mega ST(e) machines. There, the Atari was more seen as a (small) business machine, whereas in Britain it never really got over the games machine image.

It should be emphasised that Jochen Merz also sells a Mega ST adaptor, so that those having an older Mega ST can also use the QVME card (one wire needs to be soldered in that case). This is the case for me: I have a Mega 4 ST, and with the use of the adaptor, I can use the QVME card. This is the configuration actually used to write this review. **Figure two** shows the card comfortably installed inside the machine.

Of course, one will need a suitable monitor, a multisync being the best. Monitor size is also important. A 14 inch monitor can't really go much higher than the 832x416 resolution in **Figure three**, or the letters are too small for comfort. With a 17-in monitor, that should be possible, though. A normal QL monitor will not be able to display a very high resolution screen, but an ordinary VGA screen will be able to display at least 640x480 - not bad, considering how cheap VGA monitors are today.

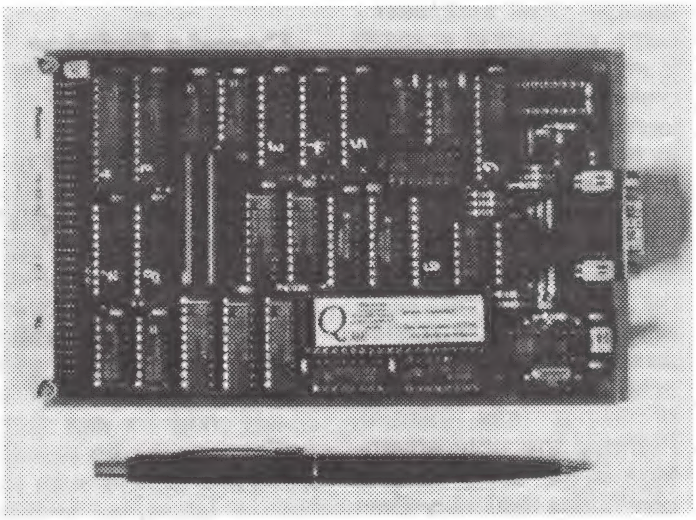


Figure one: The QVME card with its VME connector.

Large Screen

Once installed, the card must be connected directly to the monitor. Indeed, the QVME card is actually a fully-fledged **graphics card for the STe** which replaces the latter's internal circuitry. (Drivers to use it in Atari native mode are also provided, but this does not work on ST machines, one needs a genuine STe.) The reason for such a graphics card is that it can display resolutions much higher than the Atari can: resolutions up to 1024x768 are possible - even in QL mode: more than four times the native QL screen!

The QVME card does work with a normal QL monitor - but you will need to have someone make up a plug, as the card only has a 'normal' (for the PC world 15-pin plug. Even on a normal QL monitor, it is possible to get resolutions higher than the normal QL resolution.

Adapting

To adapt the card to your monitor, a whole suite of new keywords are linked into Basic. These allow you to change resolutions, vertical and horizontal frame frequencies, vertical and horizontal overscan etc...

Fortunately, these new keywords and their purposes are well explained in the manual. Initially, the screen will be set to QL compatible

resolutions and frequencies, it is thus a matter of including a few more lines of Basic in your boot file to change these. Once one has found a correct set of values, these can be preconfigured, so that even the Basic lines in the boot become obsolete.

How does one find the best values? By trial and error - increasing (for example) the resolutions ever so slightly, until finding a suitable one. This raises two interesting points:

First of all, this method seems a bit fussy: wouldn't it have been better to give a table with standard values? Perhaps the reason lies in the fact that there are too many different monitor types. Here, the manual gives a very stout warning: do not try to use frequencies higher than the ones supported by your monitor, else you can (and will!) physically damage it. The manual adds: "Take this warning seriously!" So you have been warned. This is probably why it is left to each user to select the best frequencies for his/her monitor (you will need your monitor's manual for that).

Size Changes

Second and incredibly, the display changes to the new size as soon as one types the Basic commands: it is possible to change resolution 'on the fly' - and all software continues to function! Multitasking really takes on a new meaning as it is possible to display several jobs at once. Likewise, there is now space enough to display a very large list of files on the hard disk, without having to scroll through too many of them (**see Figure four**).

When programming, those using QD are also happy, as the window of that editor can be enlarged, so that one can see many more lines at a single glance.

So what is using the card like? First of all, having a high resolution QL (after all, that is what it boils down to) is addictive! At last there is enough space for a whole row of buttons at the top of the screen (see **Figure three**), visually recalling the jobs running in my machine. It also becomes realistic to leave many window open, as somewhere part of it will show and thus can be 'picked' quickly with the mouse. With a large screen, multitasking gathers a new dimension. This enables me to integrate the computer even more into my business life. I have written a suite of programs to assist me in (or automate) most material aspects of my law office (mainly follow-up of cases, accounting, standard letters etc). Now the AtaQL reflects even more closely the way I work, compelled (through telephone calls, urgencies etc..) to switch quickly from one task to another. In that sense, I can certainly say that the larger screen is - for me - a productivity tool.

Several Things

In addition to the possibility of seeing several things at once, it also becomes possible to see more (of a list, a letter, a deed etc) at once. This, again, speeds up my work - nothing can beat the humain eye/brain combination to pick up fast salient facts from a screenfull of lines. Unfortunately, however, this latest possibility is still restricted: not all programs can make use of the large screen; only a few do. They run, but they still assume the the maximum screen size is 512x256. Some programs (such as the QPAC 2 suite, QD, Qspread and so on) allow their screen size to be altered and of course, these programs can take advantage of the large screen. With one exception, they invariably run under the pointer environment. The exception is Text87 - it will adjust to a larger screen, even though it is not a pointer environment product. For various reasons, I don't use Text87, and my biggest lament is that I don't have a wordprocessor showing me more of a page!

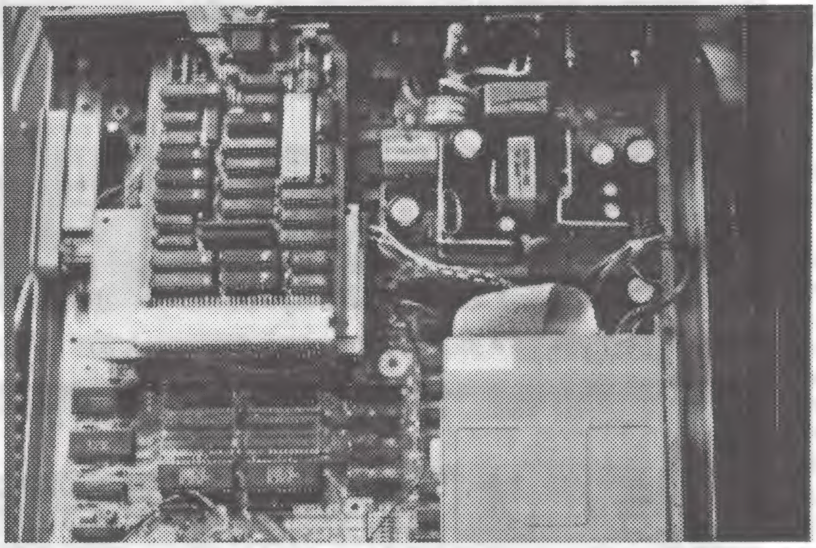


Figure two: The QVME - a QL card installed in an Atari STe.

One aspect of interest might be the screen updating. We all know by now that, in the QL, screen access is rather slow. On the Atari, this is not the case, so, compared to the QL, scrolling a page up or down in blindingly fast. With a larger screen, however, scrolling speed goes down again. This is easily explained. Suppose one scrolls the screen down by one pixel line. This means that the entire screen must be moved down one line. At a normal QL resolution of 512x256 pixels in 4 colours, the processor must thus move 512*256/4 bytes (there are 4 pixels to a byte in 4 colour mode) = 32768 bytes. When using (say) a 1024*512 resolution with the card, the processor must move 1024*512/4 bytes = 131072 bytes, i.e. four times as much a normal QL! Despite that, scrolling on an AtaQL is still fast - as fast as a normal QL!

Of course, there are some other niggles. To my mind, they are minor, but I know of others who do feel incommoded. Perhaps the most lamented is that the emulator only allows Mode4. There is no Mode8 at all. I may be wrong, but to my knowledge, the only programs using Mode8 are graphics programs, or games. I only rarely use the former - and the games I play work in Mode4. As I never use Mode8, I can honestly say that I don't miss its absence at all.

Compatibility

Also, there are some compatibility problems: no program writing direct to the screen will work - the screen address does not lie at $20000 as in the QL. It seems to be somewhere in high memory. But there again, the majority of programs writing direct to the screen are games... It must also be mentioned that some programs, even though they write directly to the screen, can be persuaded to run nevertheless: such as the Psion suite, all of which (except Easel, for obvious reasons) can run on the emulator. Indeed, in fig-

ure three, you can see Quill, being used to type this review (actually, I cheated, because the is the Quill of Turbo-Xchange, which can also be persuaded to run on the emulator).

Another potential source of trouble could be the pointer environment, which is automatically loaded into the machine. I personally think that the pointer environment is the best thing to have happened to the QL since its inception. I would actually refuse to work with a machine that didn't have it loaded, but there are some people who can't get to grips with it.

I also have an unfulfilled wish. It is clear that the card can only display four colours. It would have been very nice, however, if I could choose what these colours were! The reason for that? Well, it would be nice to be able to replace the colour green (or red) by a grey. This would not make much difference to already existing programs, but I could then use white, black and grey to achieve some neat 3D effects in programs I write myself! Unfortunately, the colours are hard-coded in the QVME card, so that is not possible.

All in all, though, these criticisms are minor. For me, the situation is simple: there is no program available on the QL which I would like to use, but which I can't use because of incompatibility problems. The emulator is in my machine to stay!

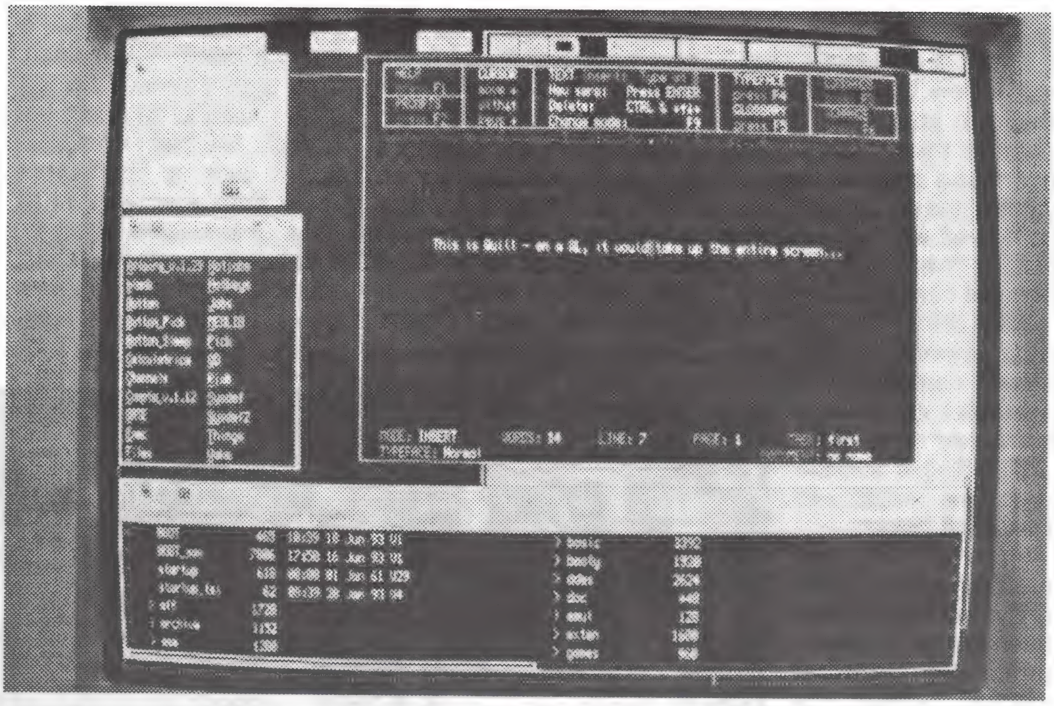


Figure three: Even a 14 inch monitor gives clear resolution at 832x416.

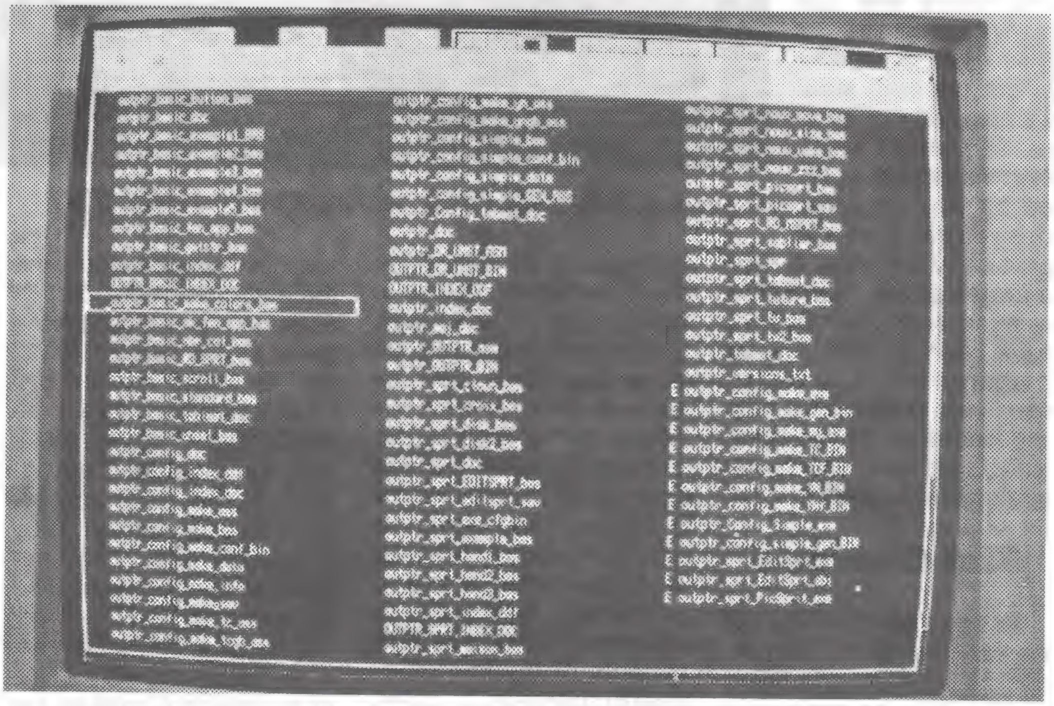


Figure four: A very large list of files can be displayed at once.

Future?

The card addresses the point I perceive to be the biggest disadvantage of the QL: the screen. Indeed, if one looks at other computers today, what most QL users find striking, is their display. Most software developed today for, notably, PCs using Windows, just looks nice. Never mind, here, whether that software works (much of it doesn't!) or does things you couldn't do with the QL (as is often, and very wrongly, stated) - the fact is that it is often pleasant to look at, heightening the interest in the program (and the computer). I firmly believe that it no longer is enough for a modern program to do what it sets out to do - it must also do it in a visually pleasing manner. This is difficult to achieve on a normal QL display! It is incredible that a modern computer (and, even after 10 years, the QL is still that!) should only have a highest resolution of 512x256 in 4 colours. The QVME card certainly puts that right!

Moreover, due to a unified interface system, modern software on other (lesser!) computers is also often more easy to use - but if every QL developer could only be made to develop under the pointer environment, the QL could also have that!

The debate over the future of the QL has been going on for some time now, and it is no accident that the QXL card should be out soon. For those QL users having Mega ST(e) computers, the QVME card could be the best way to have that future, now.

The NEW USER GUIDE

Concepts Section

Section Thirty Two

This month we begin the Hardware section of the Concepts guide.

The original Sinclair QL User Guide's Concepts section was an alphabetical mix of the trivial, the valuable and the arcane. Topics were not inter-related and the content was somewhat haphazard. The section on QDOS, for instance, is both inadequate for a machine code programmer and inaccessible to a SuperBasic programmer. The topic on peripheral expansion included the necessary disclaimer that the author did not expect QL users to make their own expansion devices based on two pages of text, but justified its presence as a way of gaining a basic understanding about the expansion mechanism. Potentially useful subjects such as arrays are dismissed in a few lines. The original Concepts section is unclear about its audience, arbitrarily assembled, and ultimately failing in its intended purpose.

The **Concepts** section in the New User Guide began last month with an overview describing the relationship between the layers of a computer system. The remaining sections will be dealt with in the following sequence: hardware, operating system (QDOS) and programming (SuperBasic). The New User Guide's reader is the intelligent non-specialist QL user who wants to make SuperBasic work but is not willing to be buried under technical minutiae. Cross references to other QL World articles are included for those who want to go further.

HARDWARE

Communications

The QL communicates with the outside world through a variety of ports, sockets and connectors. These fall into two major hardware categories: simplex devices that permit one-way communication only and duplex devices that allow two-way communications between device and computer, sometimes only in one direction at a time (known as half-duplex mode).

The simplex input connections are the keyboard (technically a peripheral), and for games joysticks (CTL1 and CTL2). The keyboard connector lies in the centre of the motherboard and is only accessible by unscrewing the body and carefully lifting the cover. Two ultra-thin ribbon connectors thread their way from the keyboard mat (for which the only benefit ever claimed was that it survived coffee spills) into two long, thin sockets. If they become detached they are a devil to force back in.

Plenty of alternative keyboards have been, and are still, available. Advanced keyboards replace the QL's original 6502 controller chip and link to any standard, PC-compatible, 102-key keyboard. This approach is a little more technically demanding than replacing one pair of ribbon connectors with another, but has the advantage that signal timing problems affecting the serial ports are avoided. These days, many QL owners use an external keyboard because they are more comfortable and offer numeric keypads and improved key-mapping (for instance, the Pause key emulates Shift-F5 to interrupt screen scrolling).

The 9-pin Atari "D" connector games joystick ports, CTL1 and CTL2, have by and large been forgotten by QL users as little use was ever made of them. Games written for two joysticks, such as Psion's *Tennis*, can also be used from a shared keyboard. Signals from the games ports can be read by KEYROW() to emulate the following keyboard presses:

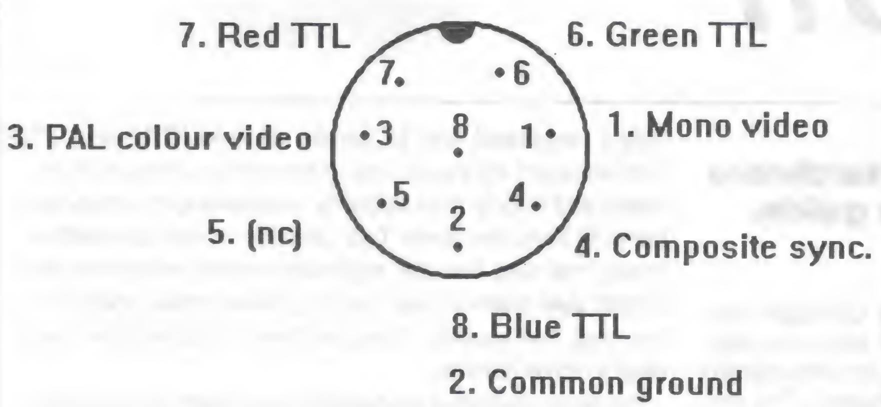
ACTIONCTL1CTL2
Joystick upup cursor keyF4
Joystick downdown cursor keyF2
Joystick leftleft cursor keyF1
Joystick rightright cursor keyF3
Fire buttonspacebarF5

The only simplex output device is the picture socket. This is a multi-purpose 8-pin DIN socket with output compatible with TTL colour, monochrome video, PAL and composite sync. **Figure one** (over the page) shows the layout of the socket as it is seen looking at the socket in its position at the back of the QL. The connector at the other end of the picture-carrying cable depends very much on the type of monitor in use. At least one type of monitor rather dangerously sports an identical 8-pin DIN socket to the QL's but with an entirely different pin-out. Connecting the cable the

wrong way round can destroy the computer's picture-generating components.

The half-duplex connections are the rom port at the rear left of the QL, the large expansion port on the left of the casing, the two network sockets near the power socket, and the serial ports in the centre of the computer's rear panel.

In the early days, the rom port was occupied by overspill hardware for the operating system and SuperBasic, universally known as the Kludge. More recent and happier times have seen the rom port used by QDOS extensions such as Lightning and SuperToolkit II, language enhancements such as MetaCompco's C compiler, and even device connectors such as a MIDI music port and a hard disk controller. One-to-many units used to be available to allow more than one external rom to be present at a time. Installation is a simple matter of removing the cover plate and inserting the rom into the waiting edge connector, but QLs are well-known to resent frequent changes of occupant in the rom port.



QL multi-format video socket

It is a rare QL that does not have something, usually from Miracle Systems, hanging from the expansion port to the left of the QL body. Some of these expansion devices have a "through port" to allow others to be daisy-chained to them. One of my QLs, for example, has an Expanderam card and a Miracle disk drive controller attached, making the system 66cm wide. The latest favourite for the expansion port is the Gold Card. Installation of any peripheral through this port is easy, although care needs to be taken: it is best to remove the lid of the computer entirely so that you can be absolutely sure that the 64 pins line up precisely with the 64 holes in the QL's socket.

The **serial ports** are, to most users, the printer ports, and are only used for simplex traffic to drive a printer. Serial cables, where files are passed bit by bit down a single piece of wire, are notoriously slow and difficult to set up. The simplest solution to most printing problems is to fit a serial to parallel converter, the device that set Miracle Systems on the road to fame. The QL thinks it is talking across a serial device using its default settings, but the signals are processed inside a small box from which they emerge on a parallel cable with each bit in its own wire. Printers receive the information synchronised byte by byte, which is faster and more secure.

The QL can accept mouse input through one of these serial ports, provided that QDOS has been extended with mouse-aware utilities such as those in the QPTR software. Other uses for the serial ports are to link to modems and to perform file transfers with other computers, notably the IBM PC, the BBC micro and the Sinclair Z88 portable. In these circumstances the port is used in half-duplex mode so that information travels in either direction.

Serial ports are deemed to be connected to DCE or DTE devices (Data Communication Equipment and Data Terminal Equipment respectively). Although in the mainframe world where serial protocols were developed it is clear which device is which, there has always been confusion with microcomputers that can at once be a DCE in relation to their printer and a DTE in relation to a mainframe connection. The QL embodies this confusion by configuring the SER1 port as a DCE and the SER2 port as a DTE, so you can take your pick. For printing, though, SER1 is the obvious choice.

Physically, the serial ports are telephone RJ11 sockets. When the QL came out this was thought by many to be an outrageous cost-cutting exercise, but the socket is very reliable and has since become widely copied by other systems, including the latest PC network structured cabling hubs. The QL's ports can be set to run at speeds of up to 9,600 baud (or transmit on 19,200 baud), but the speed cannot be set independently for each port. See the **BAUD command** in the New User Guide Keywords Section for further detail.

Useful references:

Keyboards: SQLW Feb 91 page 34, Apr 91 page 34, Jan 93 page 20
Monitor connections: SQLW May 87 page 25
Gold Card: SQLW Jun 92 page 22
RS232 serial interface: RS-232 Made Easy by Martin D Seyer published by Prentice-Hall, ISBN 0-13-783472-1

Storage Devices

The working memory of a QL is volatile: when you switch off the computer, you have 0.02 seconds to restore power before everything stored in random access memory disappears. An essential part of the computer is therefore a means of storing information somewhere where constant electrical power is not required to keep the binary information intact.

To begin with, the QL relied on **microdrives**, Sinclair's unique serial-storage format. Later improvements included a variety of floppy-disk formats and even hard disk drive units.

The microdrive cartridge contains a continuous loop of about 56cm (22 inches) of video-quality magnetic tape. The tape spools up through the centre of a drum and follows a path around the periphery of the casing before being wound back onto the drum. At the top of the casing it becomes exposed. Where the heads will make contact it is cushioned by a small felt pad on a springy metal tab. Should the tab fall off it must be replaced if the tape is to be readable. Cartridges can be made read-only by removing a plastic tab from one side of the casing. Should you change your mind, the process can be reversed by applying sticky tape to the gap.

Microdrive cartridges need to be formatted, or divided into data segments, before use. The tape is notorious for stretching over time, which quickly puts the data segments outside the tape drive's limited tolerance. Always format a tape between four and six times and reformat your tapes every year as a precaution against further stretching. Each sector holds 512 bytes and it is perfectly possible to achieve between 210 and 220 sectors on a tape, giving over 100KB of storage capacity.

Stretching is also caused by heat. Unfortunately, the microdrives are placed close to a heat radiator inside the QL's body, so one of the least safe places to leave a tape cartridge for any length of time is in a microdrive slot.

The drives themselves have a mechanically-governed electric motor where contact brushes "lift off" at around 2,400 revs per minute. The motor drives a rubber roller that pulls the tape past a capstan. Crude adjustment to the position of the reading and writing head can be made using a small screw mounted immediately below the mouth of each drive on the base of the QL. Changes should be made cautiously and tested for improvements. As a first trial, try tightening rather than loosening the screws by half a turn. Always remember how many half-turns you have made so that you can return to the original settings if necessary.

Sinclair Research seemed to have a phobia about floppy disk drives, which is why no QL rom contains diskette-aware routines. Although the drives are more expensive than microdrives, floppy disks are much cheaper than microdrive cartridges and far more reliable. Fortunately, QDOS is very flexible when it comes to device types and names and both of the expansion ports can be used to link to disk drives of some sort. The traditional way is to add a disk controller, often with additional ram on board, to the QL's expansion slot. Most QL systems have 3.5-in disk drives of various densities fitted, but early upgraders also chose to use the older 5.25-in disks - the last floppy disk format to be genuinely floppy. Floppy disks are made of the same material as microdrives, but of a stiffer gauge [and flattened into a disk] and magnetically coated on both sides. Formats are Double Density (720K), High Density (1.44MB) and Extra Density (3.2MB). A system will only accept disks formatted to a capacity compatible with both the disk drive and the disk controller. The most capable system on the market today is Miracle System's Gold Card, which copes with all these disk formats (which is not true even of your above-average PC) as well as supporting huge ram expansions and ultra-fast access to the QDOS and SuperBasic routines previously stored in slow roms on the motherboard.

While floppy disk drives arrived early and are still very popular, hard disks for the QL came late. A hard disk drive is permanently attached to the computer and stores, in the QL's case, around 32MB of data. ABC and Rebel once made controllers that attached to the expansion port. Miracle's hard disk unit slotted into the rom port and had a through-port to daisy-chain another rom. More recent hard disk interfaces, like Jurgen Falkenburg's, can be connected in a variety of ways.

Hard disks were expensive on a price per megabyte basis, and noisy, due to the spinning disk and the cooling fan. In the PC world, where 20MB programs are no longer rare, large hard disks are essential, but most QL users are better served by Extra Density diskettes. These offer the same sort of access speeds as hard disks, but are cheaper and more flexible in use. QL programs are very compact compared with PC equivalents, so a single 3.2MB diskette can store a major program and a substantial amount of related data.

Useful references:

Microdrives: SQLW Jan/Feb 86 page 36, Mar 86 page 34, Jan 89 page 33
Floppy disk drives: SQLW Jun 86 page 16, Mar 89 page 34
Hard disk drives: SQLW May 90 page 14, Sep 90 page 36
Gold Card: SQLW Jun 92 page 22

Roms, Eproms and Expanson Boards

The personality of a QL lies in its system roms, mounted on the motherboard. **Sinclair Research roms** came in pairs, but can be replaced by a single Minerva rom. The saga of the roms encapsulates all that went wrong with the QL, but that story belongs to the QDOS part of the Concepts section. The system roms for the QL fall into three categories: the early series up to and including AH, the late series of Sinclair roms from JM to MG, and the Minerva. The early roms had hardware differences which make them difficult to upgrade. All roms from JM onwards, including Minerva, are plug-compatible. In other words, you can remove one rom set and insert any of the others. Minerva looks different from the other roms because it does not fit directly into the motherboard socket but sits on a daughterboard.

Physically, the system roms look like large memory chips: the legs fit into sockets on the QL motherboard. Replacing a rom involves removing any chance of static electricity destroying the components and then levering the old chips from their sockets and firmly and persistently applying pressure to encourage the new chip or chips to take their place.

The force required can appear to be quite exceptional, but it is necessary. Great care is needed to avoid bending any of the legs.

Very early QLs could not fit their operating system and programming language into the 48K of the two roms and so needed a third rom mounted through the rom expansion port. It is doubtful that any QL has this configuration today, but one useful reminder exists of those times. To relieve the demands on limited rom space, Sinclair considered dividing the SuperBasic commands into two categories, with commands only of interest to programmers being loaded from microdrive into ram. In preparation for that move, much of the SuperBasic language was coded as extensions to the QDOS core, which means that it is relatively easy to "unlink" parts of the language and replace it with alternative commands from other sources.

Roms come in three categories: fixed, programmable and re-programmable. True roms are cheap to manufacture from a die, but cannot be re-used. Programmable roms begin life as a complete set of binary ones and a program is run to create the necessary zeroes by, in effect, breaking tracks on a matrix. Erasable re-programmable roms (Eproms) are also programmed electronically, but prolonged exposure to ultraviolet light (for instance, by taking them to a disco) can reset the rom ready for a new set of instructions. If there is a bug in the contents of either of the first two rom types, the only solution is to throw them away and start again. However, they are considerably cheaper than Eproms. Incidentally, equipment such as electronic music keyboards use a fourth category of rom, electronically erasable programmable read only memory, or Eeproms. These can be erased by an electrical charge. Although the value of such chips in the computer business is easy to see, they are extremely expensive.

Another disadvantage with all kinds of rom is access speed. Random access memory might be volatile, but it is very fast. A major part of the power of the Miracle Gold Card lies in its ability to copy motherboard roms into its own ram as part of its start-up sequence.

Random access memory is also located on the motherboard of the QL. Its chips are physically smaller than roms, but there are many more of them. Memory expansion boards containing more ram chips can be added to the QL via the 64-way expansion slot. The upper limit of ram is decided partly by QDOS and partly by the address bus, which is wide enough for a total of 16MB memory. System roms, screen maps and QDOS tables occupy part of this figure, but even so there is room within the theoretical maximum for whole suites of QL programs and their data. More realistically, the Gold Card is supplied with 2Mb of ram and the brand new Miracle QXL card has a maximum memory capacity of 8MB.

Useful references:

Sinclair rom family: SQLW Aug 87 page 18, Dec 87 page 17, Feb 89 page 19
Minerva rom: SQLW Nov 89 page 26, Sep 90 page 18, Dec 90 page 28

QL Clones

Much of the unreliability that made the QL's life so difficult at the beginning was due to the physical components. Just as the problems with QDOS and SuperBasic were resolved by evolving a family of roms, subtle changes appeared in the electronics to improve basic reliability. Most QLs were manufactured by Thorn-EMI at Feltham, miles away from where the QL was designed and developed in Cambridge. The last QLs were manufactured by Samsung of Korea and are generally acknowledged to be superior to British-built ones, although all late-model QLs are considerably more reliable than their predecessors.

Despite the changes, it takes a keen eye to spot differences in the motherboards of Sinclair-sponsored QLs. Far more dramatic shifts occurred when QDOS and SuperBasic were transported to other computers. This was initially done with emulation programs, sometimes backed up with hardware, running on the Atari ST computer, chosen because its CPU belongs to the same family as the QL's 68008. The emulation runs much faster than a real QL.

Other machines were basically a QL in another skin: the Thor, the Merlin Tonto sold by British Telecom, and the ICL One-Per-Desk. All sold surprisingly well, and were used by a wide variety of corporate clients, but did not penetrate the home market at all.

In a way, the Gold Card is a QL emulation running on a QL, because all the rom is copied into the Gold Card's own memory. The latest hardware transformation is the Miracle QXL, an IBM PC expansion card with software to turn its PC host into a true QL. Standard PCs use an Intel CPU in the 80x86 series which is quite unsuitable for QDOS. The QXL card has its own processor, the Motorola 68EC040. This is very much faster than the 68008, but shares the same machine code instruction set.

Useful references:

Crash-proofing your QL: SQLW Nov 89 page 22
The Thor: SQLW Jul 86 page 18, Jul 87 page 12
The ICL One-Per-Desk: SQLW Nov 91 page 24
Miracle QXL: SQLW Volume II Issue 6 page 8 and 15, Volume II Issue 8 page 4

32 Section Thirty Two

SuperBasic In Action

Simon Goodwin senses two new TS Services I2C analogue interfaces.

Tony Firshman has gone to a great deal of trouble to ensure that his new **I2C interfaces** for the QL are programmable from Super-Basic. This column demonstrates applications for the new Analogue interface, with working Super- Basic to play sampled sounds, convert them from other disk formats, graph sampled data and read a proportional 'analogue' joystick in two dimensions.

Up till now, QL computer signals have invariably been digital. This means they are considered to be at one of two levels, either on or off, true or false. An analogue interface allows intermediate values, like shades of grey. In this case, a byte of data corresponds to a linear scale of 256 voltages, from a preset minimum to a maximum of a few volts. Your programs can generate waves or read instruments that return varying signals.

Analogue Signals

This is not the place to review the I2C hardware; suffice it to say that it's a sturdy black box that connects to the Mark 2 Minerva via two plugs and a ribbon cable. You can connect and use several parallel or analogue interfaces at once, although the software only communicates with one at a time. I tested the Analogue version which provides four eight-bit inputs per chip, and a pair of eight-bit outputs.

The I2C interface is controlled by a tiny machine-code extension, I2C_IO, which calls new code in the Minerva rom. The I2C vector appears in all the latest Minerva roms, but you need the extra chips in the Mark 2 version (with a real-time clock) to communicate with other I2C devices.

I2C_IO lets you use the small amount of non-volatile memory in the clock chip, but the protocol is complicated and error prone, with up to four parameters and an intricate message string. The Analogue and Parallel I2C interfaces come with extra commands that keep I2C_IO at arm's length. These extras are written in SuperBasic by Tony Firshman. You can either edit and interpret the original source, or use the equivalent QLiberated code, also supplied.

I2C_IO is tersely documented at the back of the latest (green) Minerva manual. The I2C interfaces come with a 12 page manual that explains the hardware and four new extensions. READ_PAR and WRITE_PAR transfer eight bit signals to or from any parallel port.

WRITE_ADC% and READ_ADC$ move strings of values to or from the Analogue/Digital Converters (ADCs). Characters in the string are transferred

```
100 REMark I2C Analogue interface demonstrations
110 REMark By Simon N Goodwin & Tony Firshman, 1/94
120 :
130 DEFine PROCedure JOYSTICK
140 REMark QL I2C Analogue joystick input
150 x=100 : y=100 : OVER -1
160 BLOCK 10,10,x,y,2
170 REPeat loop
180   k$=READ_adc$(1)
190   IF k$="" : BEEP 1000,10 : NEXT loop
200   BLOCK 10,10,x,y,2 : ox=x : oy=y
210   x=CODE(k$(1)) : y=CODE(k$(2))
220   BLOCK 10,10,x,y,2 : REMark PRINT #0,x,y
230 END REPeat loop
240 END DEFine JOYSTICK
250 :
260 DEFine FuNction READ_adc$(address):LOCal addr%,a$
270 addr%=test_adc_addr%(address):IF addr%<0:RETurn ''
280 a$=CHR$(160)& CHR$(4)& CHR$(1)& CHR$(178)& CHR$(3)
290 RETurn I2C_IO(a$ & CHR$(156) & CHR$(255),5,addr%,1)
300 REMark 160 is send START+device, write from control
310 REMark (length 1 is preset), no STOP. First 4 is
320 REMark the ADC control to select mode of operation
330 REMark 1 will be the first acknowledged byte count
340 REMark 178 is send START+device, read 1 acknowledged
350 REMark byte to register - ie lose duff old sample.
360 REMark 3 will be the acknowledged read byte count
370 REMark 156 is read 3 acknowledged bytes to buffer,
380 REMark read one more byte to buffer without
390 REMark acknowledge, send STOP; 255 is standard finish
400 END DEFine READ_adc$
410 :
420 REMark non-zero error code on error
430 REMark $$external
440 DEFine FuNction WRITE_adc%(address,x$)
450 LOCal addr%,message$
460 addr%=test_adc_addr%(address):IF addr%<0:RETurn addr%
470 message$=CHR$(160) & CHR$(68) & x$ & CHR$(255)
480 RETurn I2C_IO(message$,0,addr%,LEN(x$)+1)
490 REMark 160 is send START+device, write from control
500 REMark buffer (length LEN(x$)+1 is preset)
510 REMark 68 is the ADC control byte, x$ is the string
520 REMark x$ is the string itself, 255 marks the end.
530 END DEFine WRITE_adc%
540 :
550 DEFine FuNction test_adc_addr%(a)
560 RETurn test_addr%(a,72)
570 END DEFine test_adc_addr%
580 :
590 DEFine FuNction test_addr%(a,m%)
600 LOCal a% :REMark Simplified by SNG
610 IF a<"0" OR a>"127":RETurn -4:REMark Out of range
620 IF INT(a)<>a:RETurn -15:REMark Bad parameter
630 a%=a&&-8:SELect a%=0,m%:RETurn m%||a:REMark OK
640 RETurn -7:REMark not found
650 END DEFine test_addr%
660 :
670 DEFine PROCedure REPLAY(file$)
680 REMark Replays samples up to 32K long from disk
690 REMark to DAC channel 1. Uses FLEN, INPUT$,NEWCHAN%
700 LOCal k$,e%,ch% : ch%=NEWCHAN% : OPEN_IN #ch%,file$
710 sample$=INPUT$(#ch%,FLEN(#ch%)) : CLOSE #ch%
720 e%=WRITE_adc%(1,k$)
730 END DEFine REPLAY
```

Listing one.

Listing two

```
100 DATA_AREA 99
110 IMPLICIT% ch
120 max=32750
130 DIM k$(max),hunk$(20),file$(42),name$(4)
140 INK 7 : INK #0,7 : PAPER #0,0
150 CLS #0 : CSIZE #0,2,0
160 REPeat try
170   PAPER 0 : CLS : CSIZE 2,1
180   PAPER 2,0,3
190   AT 0,2 : PRINT "   Amiga 8SVX IFF sample   "
200   AT 1,2 : PRINT " to I2C Analogue converter "
210   CSIZE 2,0
220   PAPER 2
230   AT 5,2 : PRINT "v1.7 © Simon N Goodwin 1994"
240   STRIP 0 : PRINT
250   CONVERT_IFF
260 END REPeat try
270 :
280 DEFine PROCedure CONVERT_IFF
290 LOCal ch,in%,size
300 REMark Converts signed 8SVX byte IFF samples
310 REMark to I2C unsigned format; uses INPUT$ & ANYOPEN%
320 REPeat loop
330   INPUT #0;"Enter IFF drive and file name:"\file$
340   IF file$="" : PRINT #0;"STOPPED" : STOP
350   in%=ANYOPEN%(file$,1)
360   IF in%>=0 : EXIT loop
370   PRINT #0;"Error " & in% & ", please try again."
380 END REPeat loop
390 HUNK name$,size
400 IF INPUT$(#in%,4)="8SVX"
410 REPeat scan
420   HUNK name$,size
430   IF name$="BODY"
440     CONVERT size : EXIT scan
450   ELSE
460     REMark Skip this hunk
470     hunk$=INPUT$(#in%,size+(size MOD 2))
480     REMark Skip nulls
490     FOR ch=LEN(hunk$) TO 2 STEP -1
500       IF hunk$(ch)=CHR$(0):hunk$=hunk$(1 TO ch-1)
510     END FOR ch
520     IF LEN(hunk$)>1
530       INK 4 : PRINT ,'"' & hunk$ & '"'
540       INK 7
550     END IF
560     IF EOF(#in%)
570       PRINT #0;"No sample body found." : EXIT scan
580     END IF
590   END IF
600 END REPeat scan
610 ELSE
620   PRINT #0;"Sample is not in IFF 8SVX format."
630 END IF
640 CLOSE #in%
650 END DEFine CONVERT_IFF
660 :
670 REFERENCE k$,k
680 DEFine PROCedure HUNK(k$,k)
690 k$=INPUT$(#in%,4)
700 PRINT "Hunk name = " & k$;
710 k=0 : FOR ch=1 TO 4:k=k*256+CODE(INKEY$(#in%,-1))
720 PRINT " Hunk size = " & k
730 END DEFine HUNK
740 :
750 DEFine PROCedure CONVERT(bytes)
760 REMark Signed bytes to unsigned, one by one!
770 LOCal ch,k%,out%
780 IF bytes>max
790   PRINT \"File too long - truncating to " & max
800   bytes=max
810 END IF
820 k$=INPUT$(#in%,bytes)
830 PRINT \"Converting sample ... ";
840 REMark OPEN #5,scr_512x256a0x0
850 FOR ch=1 TO bytes
860   k%=CODE(k$(ch))
870   IF k%>127 : k%=k%-128 : ELSE k%=k%+128
880   REMark PLOT #5,i% && 511,k%
890   k$(ch)=CHR$(k%)
900 END FOR ch
910 REMark CLOSE #5
920 PRINT "OK"
930 INPUT #0;"Enter output drive and file name:"\file$
940 out%=ANYOPEN%(file$,2)
950 IF out%>=0
```

one by one over the I2C interface, at a rate of around 10,000 eight-bit bytes per second.

Simplified source for WRITE_ADC% and READ_ADC$ appears in the first listing.The originals used clever SuperBasic to check parameters, but I've replaced this with simpler stuff to catch most cases.

The function TEST _ADDR% checks that an I2C port address is valid, using interesting bitwise operations. The extensions are carefully designed to detect mistakes before they reach the I2C ports, and return meaningful error-codes.

The I2C address contains two parts in one byte. The bottom three bits are the unit number, while the other bits identify the type. Thus you can have up to eight interfaces of each type. When you call READ_ADC$ or WRITE_ADC% you can pass the full eight-bit address or a simple unit number 0 to 7, in which case the code sets the other bits to suit the type. The 'bitwise or' operator || [two vertical bars] combines the type, M%, with the address in A. a%=a&&-8 returns zero if a is less than 8; otherwise it sets A% to the value of A, with the bottom three bits cleared.

The value -8 corresponds to 11111000 in eight-bit binary. The SELect on A% ensures that the type must either be zero or match M%, which is 78 for Analogue (ADC) ports or 56 for PAR ones.

I put the range check values in quotes on line 630, so errors are unlikely even if parameter A is passed as a string, rather than a number. The quotes ensure that the parameter is coerced into a string before the comparison takes place. For yet more reliability, use UNSET from DIY Toolkit Volume P to ensure that the parameter has a value.

Demonstrations

The JOYSTICK procedure contains a small loop that lets you move a block cursor around the screen with an analogue joystick. The cursor is drawn in OVER -1, so that it complements the background colour and vanishes when drawn a second time. Type OVER 0 to cancel this if subsequent PRINT commands give messy results.

The demonstration runs as a loop, reading the inputs on port one and converting the resultant characters into co-ordinates for BLOCK. Remove the REMark from the middle of line 220 if you want co-ordinates displayed in window #0 as the block moves. There's little risk of an 'out of range' error as new Minerva roms clip BLOCKs to fit any window.

I made up a simple lead to connect the two potentiometers of a £5 Amstrad PC joystick to a pair of I2C analogue inputs. This could replace a mouse, as long as eight bit (MODE 8 pixel) resolution is good enough for your purposes. The advantage is that results are repeatable. Any particular position of the stick will correspond to a position on the screen.

As you move the stick the cursor jumps directly from place to place, in proportion to the movement. This is far more precise than a switched digital joystick, like the sort that plugs into QL Ctrl ports, which can only signal 'up', 'down', 'left' or 'right', with no indication of proportion or speed.

Depending on your graphics routines you reverse the Y co-ordinate and scale the results from

```
960   PRINT "Saving " & file$ & " ... ";
970   PRINT #out%,k$;
980   CLOSE #out%
990   PRINT "OK"
1000 ELSE
1010   PRINT #0;"Error " & out% & ", file not saved."
1020 END IF
1030 PAUSE 75
1040 END DEFine CONVERT
1050 :
1060 DEFine PROCedure S
1070 SAVE FLP1_IFF_TO_I2C_BAS
1080 END DEFine S
```

READ_ADC$ up or down. Most sticks have knobs that limitf travel.

Often joysticks use a small part of the possible potentiomenter movement, so they do not return the whole voltage range. If you need maximum precision, with values from 0 to 255 rather than (say) 80 to 160, you might need three resistors or a couple of presets to set appropriate limits for the interface. If these are set too close together the stick will only make a difference for part of its travel, and will stick at 0 or 255 before it reaches the end-stop.

More Precise

PROCedure REPLAY uses the function WRITE_ADC% to play a sampled sound. The graph shows the wave - a sampled recording of 'Rimmer' from the BBC space-ship 'Red Dwarf' saying 'Marvellous' with characteristic enthusiasm. NEWCHAN% is from DIY Toolkit Volume R. It returns a free channel number.

The 32K string length limit means that each utterance is limited in length to about 3 seconds, but you can fire them out in rapid succession. There's a short pause as SuperBasic hefts each new string into place, but you could create near-continuous speech by assembling and replaying 'phoneme' strings.

The sample rate is better than telephone specification but worse than AM radio. Speech is clear enough, while music sounds recognisable but a little distorted. It is important to use the full eight-bit range when recording samples, without overloading. To check this, display the sample as a graph.

The graph uses two dots (sample levels) in each column, which means each width of the mode4 screen corresponds to about one tenth of a second, at the fixed I2C rate of 10K per second.Samples might not be recorded at this rate - possibilities range from around 2,000 per second for toys to 44,100 for CDs - so they may sound slow or fast when replayed via I2C. Once the wave is in digital form you can change the length and pitch by interpolating extra values to slow down a fast sample, or averaging them in a sample that plays too slowly.

The two traces follow the 'M' sound at the beginning of the word. The middle trace was drawn first. The level is limited by the louder 'ah' sound that comes later. The whole sample uses about 17K.

PROCedure GRAPH plots the wave. The PUT #3\0 in line 150 is only needed if you want to re-plot without re-opening the file. Lines 170 and 180 use BLOCK efficiently to plot a fine millisecond grid. Change the limit of the FOR T loop if you want more of the sample overlaid on each screen. Lines 220 and 230 ensure that each trace appears in a new colour, starting with green, then white, then red, and back to green. Pixels are lit by the DIY Toolkit PLOT function, from graphics Volume G. If you don't mind a slower, flickering display you could replace PLOT with a call to BLOCK, like this:

270 BLOCK #4,1,1,j, CODE(k$(i)),colour

Sinclair's BLOCK routine is clumsy if you're trying to light individual pixels, but at least it's not as eccentric as POINT. In QL Mode8 and Thor Mode12 you need to change the width (second parameter) to 2 or the dot will not appear.

Flicker appears because each BLOCK is drawn as a short line and clipped on either side to fit. The colour must be specified each time; move the IF test on line 230 to 295 if you want the same colours with BLOCK as PLOT.

Eight-bit samples are widely available from PD libraries; the 'Rimmer' sample came from an Amiga cover-disk. Sample formats vary; some are signed values from -128 to 127, others (like the I2C ones) are unsigned bytes, and most are packed or padded in some way.

Decoders

QL users have easy access to these files via the PD Amiga Qdos QL_HANDLER, DiscOver and similar utilities, so I've written programs to decode Amiga IFF (Interchange File Format) '8SVX' and some Microsoft .WAV files. Many utilities on Mac, PC and other computers support these formats.

The second listing is my IFF file Converter, written in SuperBasic. You need the IFF file on a QL drive first; the program converts that sample into the I2C format used by REPLAY and WRITE_ADC%.

The Turbo/DIYT extension INPUT$ is used in REPLAY and the IFF Converter to read a sequence of characters from a file in one go. If you lack INPUT$ and are in no hurry you could build a string with successive calls to INKEY$. FLEN is in Toolkit 2, PD Toolkit, Thors and most disk roms.

IFF files consist of a sequence of 'hunks' with four character names and 32 bit sizes at the start of each. CONVERT_IFF looks for the 8SVX block that contains the sample, skipping or reporting others en route. IFF supports graphics, text and animations as well as sounds; Ergon's Open World utility converts IFF graphics to QL modes.

PROCedure CONVERT changes each byte from signed to unsigned format. This conversion is accelerated amazingly much by Turbo, which is why I've supplied TF Services with a compiled version for I2C users. The initial DATA_AREA and IMPLICIT% directives instruct the Turbo compiler. Remove the REMs on lines 840, 880 and 910 to see a rough graph as the sample is converted.

There's lots more to be done with the Analogue interface and a little SuperBasic. You could keep tabs on your your environment with temperature and light sensors, ranging from old-fashioned thermistors and light-dependent resistors to the latest active sensors. You could control a robot, record speech messages for digital replay, in a QL-controlled answer-phone, or synthesise waves of sound or patterns of light with simple formulae.

Alas, I2C is not for everyone. You need the Mark 2 Minerva, and a little knowledge of electricity, to make use of any I2C interface. The programming is simple and exciting, because these ports let your QL interact with the intricate world outside, where a little SuperBasic can go a long way.

Listing three.

```
100 DEFine PROCedure GRAPH
110 REMark Uses INPUT$ and PLOT (DIY Toolkit)
120 REMark + PUT and FLEN (PD/SuperToolkit)
130 MODE 4 : red=2 : green=4
140 OPEN_IN #3,"flp1_marvelous_QL"
150 PUT #3\0
160 OPEN #4,"scr_512x256a0x0"
170 FOR y=8 TO 255 STEP 10 : BLOCK #4,512,1,0,y,red,0
180 FOR x=6 TO 511 STEP 10 : BLOCK #4,1,256,x,0,red,0
190 colour=red
200 FOR t=1 TO 2
210 REMark For more scans use T=1 TO FLEN(#3) DIV 1024
220   colour=colour+red : INK #4,colour
230   IF colour>green : colour=0
240   k$=INPUT$(#3,1024)
250   FOR i=1 TO 1023 STEP 2
260     j=i DIV 2
270     PLOT #4,j,CODE(k$(i))
280     PLOT #4,j,CODE(k$(i+1))
290   END FOR i
300 END FOR t
310 CLOSE #4
320 CLOSE #3
330 END DEFine GRAPH
```

QL Scene

Miracle Plan to Take On Software Too

After a delay since last November, while a number of small bugs were sorted out, Miracle Systems have made a lot of progress with software for their PC-colonising QL card QXL, and are hoping to have a **new release for users** by the time this news item is published.

Said Stuart Honeyball: "The disk formatting software should be fixed and the compatibility of the 8-bit slot have been sorted out. We have had one release from Tony (Tebby, QL operating system author) in the meantime which we didn't issue because there were a few things that needed some more work. They were mostly fairly minor." The SuperBasic is also expected to be ready before too long, but there is no date yet.

Stuart sees Miracle moving more and more into producing their own software as time goes on. "We need to move the QL into multi-processing", says Stuart, laughing, "I have seen the future of the QL and it is multi-processing!" Make no doubt about it, he's in earnest.

In the meanwhile, Miracle are also working on a **new version of the Gold Card.** True to their regular policy, Stuart did not want to say more about that until he can give a clear schedule for it, and asked us to ask him again in two or three weeks. Rumour, though, is whispering that the new version will be even faster than the present one.

Miracle's show calendar after February includes the ever-popular Quanta Bristol Workshop on 27th March (contact Quanta), the Quanta AGM on 23/24th April in Tynemouth, Northumberland, and another visit to Newport, Rhode Island, USA, for May 14th, where they hope to show the first version of their SCSI card. Miracle have a full diary for January and February, including Edinburgh on 12th and Bielefeld on 19th **(See QL Scene).** They came back from Manchester/Blackrod disappointed by attendance - "not enough publicity", said Stuart, and QL World can only agree.

NEW PUBLIC DOMAIN COMMS

SJPD Software have now added to their disk collection **C68 Compiler V4.01** from Dave Walker, plus two new communications packages from Jonathan Hudson: QeM Release 4.00 and QTPI (Disks SJS 44 and 77). QeM 4.00 offers full ANSI colour graphics mode, VT52 and VT100 terminal emulation, and it also supports Tandata's QConnect and Hayes Compatible Modems, along with a host of other features. QTPI runs under the Pointer Environment for those who already have the Pointer Environment, and offers all the features of QeM, plus others.

Also new to the collection is **QLfax** and **ATP Offline Reader** (SJS 74), both in the Beta Test versions at the time of writing. Steve warns: "They are in their infancy, but have reached a stage where they are almost stable (I hope!)." Also new is the demo version of LineDesign, which is also a Pointer Environment program, and has all the features of the general release of LineDesign, except, of course, that you cannot save and print with it.

In the SJPD list, there is now **Zip and Unzip 3.0** (file compressor), **QSpread Demo** (Pointer Environment spreadsheet demo), **Archive to Pipedream** Conversion Program (for Cambridge Z88 users), **Deskjet Envelope Printer** (for Deskjet 500) and **Indexer Demo**.

Steve's well-known literary collection now also features works by Jane Austen, Bram "Dracula" Stoker, Mary "Frankenstein" Shelley, Emily Bronte, Cervantes, Charles Darwin and Chaucer. Who says computer buffs don't read?

All disks are £1.75 inclusive of media and return postage, or £1 if user supplies disk and return postage.

SJPD also supplies second-user QL hardware and software, and back issues of QL magazines.

In particular Demo versions of commercial software are a very good deal if you want to check them out before deciding if they might add to your QL life. Steve's disk-catalogue is continually updated and contains some PD software too. It is free to anyone who sends a disk, return postage and a self-addressed label. Program versions are continually updated.

SJPD are at 36 Eldwick St., Burnley, Lancs BB10 3DZ. Tel. 0282 451854.

Famous Meters Hit 70 Years

Electronics and electrical experts will be pleased to hear that the well-known **Avometer,** the original multimeter, has just celebrated its 70th birthday. That's older than many of us. The "big black box" has been around since 1923, and is still going strong.

How many computers do you think will be able to say that in the year 2050?

If you think that you can persuade the family to buy you a new one for Christmas (or even next Christmas), ask for information from **Avo International, Archcliffe Road, Dover, Kent CT17 9EN, UK. Tel. 0304 202620.**

Jochen Merz has told us that the Pointer Environment is indeed supplied with their new multitasking chess program Black Knight..

Beginners' Machine Code

In part 8, Alan Bridewell sets the code to trace and manipulate pixels.

At the start of this series I said that I would illustrate all the ideas with routines that would produce something on the screen. Initially, this only needed a knowledge of the addresses of the screen ram, without any understanding of how it was organised. Later, it was necessary to realise that the screen ram was organised in words (pairs of adjacent bytes) which held the data for a set of four or eight consecutive pixels, depending on whether the QL is in mode8 or mode4.

There is normally no need to understand more than this because the complicated job of drawing lines or printing text on the screen is done by Qdos routines, which we can use in our programs without having to know what they are or how they actually work.

Pixel Printing

However, there is one type of problem where the programmer really needs to know exactly how the information for the colour of each pixel is stored. (This is true regardless of the language being used, including SuperBasic.)

There are programs where the programmer needs to find **the colours of individual pixels on the screen**. The classic example is the screen dump routine for a printer. For a colour printer, the program must first identify the pixel colour, before sending data to the printer to print a dot of that colour. For a monochrome printer the program must send data for a pattern of dots to produce a suitable grey tone for each colour. I am not really concerned here with what the program does with the data, so much as how it gets that data in the first place.

I am not going to use this to produce a screen dump routine. Firstly, I said I would make the program do something ON the screen. Secondly, different printers require different data, and anyway, not everyone has a printer. However, if you do have a

```
; ************************************************************************
;                        OPEN A CON CHANNEL
; ************************************************************************
;
.OPEN              LEA.L     CON,A1      ; CON TABLE POINTER IN A1
                   MOVEA.W   $C6,A2      ; UT_CON VECTOR IN A2
                   JSR       (A2)        ; OPEN CONSOLE CHANNEL
.STORE             LEA.L     ID,A1       ; CHANNEL ID STORE IN A1
                   MOVE.L    A0,(A1)     ; STORE CHANNEL ID

; ************************************************************************
;                       WRITE TEXT IN CON WINDOW
; ************************************************************************
;
.PTEXT             LEA.L     ID,A1       ; CHANNEL ID STORE IN A1
                   MOVEA.L   (A1),A0     ; CHANNEL ID IN A0
                   LEA.L     TEXT,A1     ; BASE OF TEXT IN A1
                   MOVEA.W   $D0,A2      ; UT_MTEXT VECTOR IN A2
                   JSR       (A2)        ; PRINT TEXT
;
; ************************************************************************
;                     FETCH A LINE OF CHARACTERS
; ************************************************************************
;
.FETCH             LEA.L     ID,A1       ; CHANNEL ID STORE IN A1
                   MOVEA.L   (A1),A0     ; CHANNEL ID IN A0
                   LEA.L     BUFFER,A1 ; BUFFER ADDRESS IN A1
                   MOVEQ     #2,D0       ; #IO_FLINE IN D0
                   MOVE.W    #BUFLEN,D2 ; BUFFER LENGTH IN D2
                   MOVEQ     #-1,D3    ; INFINITE TIMEOUT
                   TRAP      #3
; THIS LEAVES THE LENGTH OF THE STRING (INCLUDING THE LF) IN D1.
; IF WE WANT TO USE THIS STRING AS A CHANNEL NAME, WE NEED THE LENGTH OF
; THE STRING (WITHOUT THE LF) JUST BEFORE THE STRING ITSELF (WHICH IS NOW
; STORED IN THE BUFFER). SO:-
; SUBTRACT 1 FROM THE STRING LENGTH....
;
                   SUBQ.L    #1,D1       ; SUBTRACT 1 FROM D1
;
; ....AND PUT THE RESULT IN BUFPOS, JUST IN FRONT OF BUFFER
                   LEA.L     BUFPOS,A0 ; LOAD BUFPOS IN A0
                   MOVE.W    D1,(A0)     ; PUT CONTENTS OF D1 IN BUFPOS
;
; ************************************************************************
;                           OPEN THE FILE
; ************************************************************************
;
                   LEA.L     BUFPOS,A0 ; BUFPOS IN A0
                   MOVEQ     #1,D3       ; OPEN AND OLD, SHARED FILE
                   MOVEQ     #-1,D1      ; -1 IN D1 MEANS 'THIS JOB'
                   MOVEQ     #1,D0       ; #IO_OPEN IN D0
                   TRAP      #2
                   TST.L     D0          ; IS THERE AN ERROR?
                   BEQ.S     GOT_FILE    ; IF NOT, CONTINUE..
;
; ...ELSE...
;
                   MOVEA.W   $CA,A2      ; ERRO IN A2
                   JSR       (A2)        ; PRINT ERROR MESSAGE IN #0
                   JMP       PTEXT       ; REPEAT PROMPT
;
; STORE FILE CHANNEL I.D WHICH IS NOW IN A0
;
.GOT_FILE          LEA.L     FILE,A1     ; ADDRESS OF FILE IN A1
                   MOVEA.L   A0,(A1)   ; CONTENTS OF A0 INTO ADDRESS IN A1
;
; NOW LOAD THE FILE INTO THE SCREEN RAM
;
                   MOVE.L    #$8000,D2 ; FILE LENGTH IN D2
                   MOVEA.L   #$20000,A1   ; LOAD ADDRESS IN A1
                   MOVEQ     #-1,D3        ; INFINITE TIMEOUT
                   MOVEA.L   FILE,A0     ; FILE CHANNEL I.F IN A0
                   MOVEQ     #$48,D0     ; FS_LOAD IN D0
                   TRAP      #3
```

printer, and if you understand this article, and understand your printer manual, you should be able to have a go at producing a screen dump routine of your own.

Screen Ram

So how exactly is the QL screen ram organised?

Each row of pixels on the screen requires 128 consecutive bytes (64 consecutive words) to store the data. This is true regardless of whether we are using mode4 or mode8. In mode4, each word holds the data for eight consecutive pixels, making 512 pixels in a row. In mode eight, each word holds the data for only four consecutive pixels, making 256 pixels in a row. Since each mode8 pixel is twice as wide as a mode4 pixel, 512 mode4 pixels and 256 mode8 pixels occupy the same width of screen.

The difference in the number of pixels that each word can hold arises asI will describe.

In mode4, there are only four states each pixel can be: black, red, green or white. Data involving four possible values can be held by two bits (binary digits), because they can have four combinations, 00, 01, 10 and 11. Now in a word (two bytes) there are eight pairs of bits, so a word can hold the data for eight pixels.

In mode8, there are sixteen states each pixel can be: one of eight colours in two possible states, flashing or not flashing, making sixteen states in all. I will leave you to work out that sixteen possible values requires four bits, and as a word contains four lots of four bits, it can only hold data for four pixels. The entire screen contains 256 rows of pixels in either mode, so requires 256 times 128 bytes of ram, which comes to 32768 bytes.

Odds and Evens

Each word in the ram consists of an even address and the following odd address. The total number held by the word is worked out from the numbers in the two bytes as follows. The number in the even bytes is multiplied by 256 and added to the number in the odd byte. This sounds complicated until you see how it works out in binary numbers. Also, since it is the individual bits which carry the data, it is much clearer why a particular value has the effect it has iwhen we use binary numbers.

256 in decimal is 100000000 in binary, and you multiply a binary number by 100000000 by adding eight zeroes. For example, if the even bytes holds 11110000 and the odd byte holds 1010101010, the word will hold 1111000010101010 because:

(11110000 x 100000000) + 10101010 =

1111000000000000 + 10101010 =
1111000010101010

In other words, we get the entire number by simply putting the two numbers together, even before odd.

How are the colours represented by this? Let us look at mode4 first.

The even byte gives the greens and the odd byte gives the reds. Each bit read from left to right gives the row of 8 pixels, again, from left to right. A 1 means the colour is set, and a 0 means it is cleared. If both are set we get white. If neither are set we get black. It is clearer what is going on if we put the bytes one under the other, so the two bits affecting the same pixel are together. So in the example above we get:

EVEN (GREEN) 11110000
ODD (RED) 10101010

The first pair of bits are both set, so we get white. In the next pair, only the even bit is set, so we get green. Following across all the bits we can see that this word will give a row of pixels:

WHITE, GREEN, WHITE, GREEN, RED, BLACK, RED, BLACK.

Mode8 is more complicated. The even byte gives greens and flashings with pairs of adjacent bits. The odd byte gives reds and blues with pairs of adjacent bits. So taking the same example again:

```
;
; ******************************************************************
;                 CONVERT SCREEN TO MONOCHROME
; ******************************************************************
;
                MOVEA.L   #$20000,A1 ; ADDRESS IN A1
                MOVE.W    #$FF,D5    ; ROW COUNTER
.ROWLOOP        MOVE.W    #$3F,D6    ; WORD COUNTER (IN ROW)
.WORDLOOP       MOVE.W    (A1),D1    ; CONTENTS INTO D1
                MOVEQ     #$7,D2     ; 8 BITS IN BYTE
                MOVE.W    #$8080,D3  ; TEST BITS
.BITLOOP        MOVE.W    D3,D4      ; COPY TEST BITS
                AND.W     D1,D4      ; COMPARE WITH D1
                BEQ       BLACK      ; IF ZERO, THEN BLACK
                CMP.W     D3,D4      ; IS IT THE SAME AS TEST BITS?
                BEQ       WHITE      ; IF SO, THEN WHITE
                MOVE.W    D3,D4      ; RECOPY TEST BITS
                AND.W     D1,D4      ; COMPARE WITH D1
                CMPI.W    #$FF,D4    ; WHICH BIT IS SET?
                BMI       RED        ; IF LSB BIT THEN RED
                BRA       GREEN      ; ELSE GREEN
.RET            ROR.W     #1,D3      ; NEXT BITS
                DBRA.W    D2,BITLOOP ; AND LOOP
                ADDA.L    #2,A1      ; INCREMENT TO NEXT WORD
                DBRA.W    D6,WORDLOOP ; NEXT SCREEN WORD
                LEA.L     YORIGIN,A3
                ADDI.W    #1,(A3)    ; INCREMENT YORIGIN
                LEA.L     XORIGIN,A3
                MOVE.W    #0,(A3)    ; RESET XORIGIN
                DBRA.W    D5,ROWLOOP ; NEXT ROW
                LEA.L     XORIGIN,A3
                MOVE.W    #0,(A3)    ; RESET XORIGIN
                LEA.L     YORIGIN,A3
                MOVE.W    #0,(A3)    ; RESET YORIGIN
;
; ******************************************************************
;                 CLOSE THE CHANNELS
; ******************************************************************
;
                LEA.L     FILE,A1    ; ADDRESS OF FILE CHANNEL I.D.
                MOVEA.L   (A1),A0    ; INTO A0
                MOVEQ     #2,D0      ; IO_CLOSE IN D0
                TRAP      #2         ; CLOSE FILE
                LEA.L     ID,A1      ; ADDRESS OF CONSOLE CHANNEL I.D.
                MOVEA.L   (A1),A0    ; INTO A0
                MOVEQ     #2,D0      ; IO_CLOSE IN D0
                TRAP      #2         ; CLOSE CONSOLE
                RTS                  ; RETURN TO SUPERBASIC
;
; ******************************************************************
;                 PRINT BLOCK SUBROUTINE
; ******************************************************************
;
.BLACK          MOVEM.L   A0-A7/D0-D7,-(A7) ; SAVE REGISTERS
                MOVEQ     #0,D1      ; BLACK
                BRA       DRAW
;
.GREEN          MOVEM.L   A0-A7/D0-D7,-(A7) ; SAVE REGISTERS
                MOVE.B    #4,D1      ; GREEN
                BRA       DRAW
;
.RED            MOVEM.L   A0-A7/D0-D7,-(A7) ; SAVE REGISTERS
                MOVE.B    #224,D1    ; RED
                BRA       DRAW
;
```

```
.WHITE          MOVEM.L   A0-A7/D0-D7,-(A7) ; SAVE REGISTERS
                MOVE.B    #212,D1      ; WHITE
;
.DRAW           LEA.L     ID,A1        ; CHANNEL ID IN A0
                MOVE.L    (A1),A0
                LEA.L     BLOCK,A1     ; BLOCK BASE IN A1
                MOVEQ     #$2E,D0      ; SD_FILL IN D0
                MOVEQ     #-1,D3       ; INFINITE TIMEOUT
                TRAP      #3
                LEA.L     XORIGIN,A3
                ADD.W     #1,(A3)      ; INCREMENT XORIGIN
                MOVEM.L   (A7)+,A0-A7/D0-D7 ; RESTORE REGISTERS
                BRA       RET          ; RETURN TO MAIN ROUTINE
;
; *********************************************************************
;                PARAMETERS, DATA AND DATA SPACES
; *********************************************************************
;
; SPACE RESERVED FOR CONSOLE CHANNEL PARAMETERS
;
.CON            DC.W      $0000        ; BORDER COLOUR & WIDTH
                DC.W      $0007        ; PAPER & INK COLOUR
                DC.W      $0200        ; WIDTH
                DC.W      $0100        ; HEIGHT
                DC.W      $0000        ; X ORIGIN
                DC.W      $0000        ; Y ORIGIN
;
; SPACE RESERVED FOR CONSOLE CHANNEL I.D.
;
.ID             DC.L      $00000000 ; CONSOLE CHANNEL ID
;
; SPACE RESRVED FOR BLOCK DEFINITION
;
.BLOCK          DC.W      1            ; 1 PIXEL WIDE
                DC.W      1            ; 1 PIXELS HIGH
.XORIGIN        DC.W      0            ; X ORIGIN OF BLOCK
.YORIGIN        DC.W      0            ; Y ORIGIN OF BLOCK
;
; SPACE RESERVED FOR FILE CHANNEL I.D.
;
.FILE           DC.L      $00000000 ; FILE CHANNEL I.D.
;
; TEXT TO BE WRITTEN IN THE WINDOW
;
.TEXT           DC.W      $0C          ; NUMBER OF CHARACTERS
                DC.B      "FILE NAME ? "  ; CHARACTERS
;
.BUFLEN         EQU       100          ; LENGTH OF INPUT BUFFER
;
.BUFPOS         DC.W      0
;
.BUFFER         EQU       *
;
; *********************************************************************
```

EVEN (GREEN + FLASH) 11110000
ODD (RED + BLUE) 10101010

The first bit in the even byte is set, showing green, and the first bit in the odd byte is set, showing red. Green and red together give yellow. The second bit of the odd byte is also set, showing flashing. So the left pixel is yellow flashing. Following across the pairs of adjacent bits in each byte we can see that this word will give a row of pixels:

YELLOW(FLASHING), YELLOW(FLASHING), RED, RED

This all looks a bit daunting at first, but a bit of familiarity will soon make it seem fairly straightforward. Probably the best thing to do at this stage is to try poking some values in the screen ram, and try to work out why it produced the effect it did. If you go back to the listings in part 5 of this series, you will find them quite useful for this exercise.

ANDing

Knowing how your QL converts ram data into pixels is all very well, but it is no use unless your program can determine whether individual bits have been set, and then take appropriate action. There are several ways of doing this, but a couple of instructions are probably all you need for most applications.

In part 5 of this series we covered the AND instruction. This carries out a bit by bit comparison of two numbers (which can be bytes, word or longword in size), and gives a 1 if and only if both bits are 1. For example, if we AND the binary numbers 10101010 and 10000000, we get 10000000, because the left bit of both numbers is set, and is the only one set.

This gives us a mechanism for finding out if a particular bit of a number is set. We simply AND it with another number with only that bit already set. If the result has that bit set, we know the original number had that bit set set. Using this we can easily analyse each bit in turn.

We are not restricted to analysing one bit at a time either. For instance, suppose we are working in mode4. If we AND a word in the screenramwith the word 10000001000000 we can find out if the first bit in both the even and odd bytes are set. If the result is all zeros we know that both bits are clear, and the left pixel of the word is black. If the result is not zero, further analysis will tell us if it is green, red or white.

Although we could make do with this alone, to analyse a whole word in mode4 would require eight repetitions of almost identical code, with only the test word being changed. Our program could soon become very long! We could use the same code over again if we had an easy way of altering the test word to test for consecutive bits in turn automatically. To do this we can use one of a set of instructions called shifts and rotates.

RORing

I am only going to deal with one of these, the one which is useful in this particular problem. The instruction is ROR, which stands for "rotate bits to the right". In this instruction, the bits of the number form a loop with the carry bit in the status register. When the instruction is carried out, all the bits in the number move one place to the right. The right hand bit goes into the carry bit of the status register, and the contents of the status register goes into the left hand bit of the number. Repetition of this instruction will eventually bring the original number back into place, hence the term "rotate".

What makes the ROR instruction useful here is this. If we ROR the test word between each AND instruction, we can use the same AND instruction over again in a loop to test each bit in turn, instead of having long repetitive code with just one line slightly altered.

In order to demonstrate these ideas, I have produced **Listing one**, a routine which will load a screen from a file, and then convert it from colour to monochrome in mode4. When I started experimenting with this routine, I intended to

```
Listing 2

100 z=RESPR(512)
110 LBYTES flp1_Listing1_code,z
120 OPEN#3,con_512x256a0x0
130 REPeat loop
140 CLS#3
150 PRINT#3,"Process a screen file? (Y/N)"
160 a=CODE(INKEY$(#3,-1))
170 SELect ON a
180 = 89,121:CALL z
190 = 78,110:STOP
200 END SELect
210 PAUSE -1
220 END REPeat loop
```

convert red and green into dark and light grey, leaving black and white the same. However, the stipple combinations for these tend to produce a stripey effect, which is not very good. In the end, I opted for converting red into a green/black chequered stipple, and white into a green/white chequered stipple, leaving black and green the same. This produces an acceptable black and green monchrome effect. After you have typed in the routine, it is easy to alter these to see if they get better.

The listing is quite long, but most of it involves a recycling of ideas used previously, so will not require much explanation.

We start by opening a console channel to input the screen file name. The window occupies the entire screen, so that it can also be used to put new pixels on the screen. Next, we write a prompt in the window asking for the name of the screen file. A line of characters is fetched from the keyboard and stored in a buffer. An attempt is then made to open a file, using this string as a file name (including drive). If this is unsuccessful for any reason, an error message is printed, and the prompt for a file name is repeated. If this is successful, the file is opened, and the contents loaded into the screen.

The Procedure

This has all been covered in previous parts of this series. The spaces required for parameters, buffers, text and channel IDs are towards the end of the listing.

Having loaded our screen file, we now convert it to monochrome, which is the main part of the program. It is probably best to read this part while following the listing.

First we move the start address of the screenram($20000) into register A1, and move $FF into register D5 as a row counter. Neither of these need resetting during the routine, so they are outside the main loop. The main loop, called ROWLOOP starts with moving $3F into register D6 as a counter for the words in the row. This will need resetting each time we start to process a new row, which is why it is in that position. The first embedded loop, called WORDLOOP, starts by moving the contents of the word whose address is in A1, into register D1, ready for processing. This processing involves looping through the eight bits in the even and odd addresses of the word together, so we move 7 into register D2 to act as a bit counter.

Next, we move the test word (the one we are going to AND with the data) into D3. As we are going to start by testing the far right bit in the two bytes, we move the binary number 1000000010000000, which is the same as the hex number $8080. Your assembler may allow you to use binary notation in the listing, in which case do so, because it makes it much clearer why that number is being used.

We now come to the inner loop, called BITLOOP. Becauase the AND instruction might alter the test word, we have to make another copy of it to do the actual testing, so we start the loop by moving the contents of register D3 into register D4, which will actually be used in the testing.

We now AND the word in D1 (which contains the contents of our screenramaddress) with the test word in D4, leaving the result in D4. If the result is zero, the pixel is black, and we branch to an address to process a black pixel.

If the pixel was white, the test word would be unaltered. We compare it with D3 to see if they are still the same, and if they are, we branch to process a white pixel.

If the pixel was neither black nor white, it means that either the bit in the even or the odd address was set, but not both or neither. To find out which, we need to repeat the first two lines of BITLOOP to test the two bits again. The result of this test is compared with $FF.

Red and Green

Now if the pixel is red, then the AND will result in a number less than $FF, and the comparison will result in a minus number (remember, CMP involves subtracting). In this case we branch to process a red pixel.

If each of these does not result in a branch, the pixel must have been green (the only other possibility), so we

```
Listing 3

100 REMark Sinclair QL World HEX LOADER v 3
110 REMark by Marcus Jeffery & Simon N Goodwin
120 :
130 CLS: RESTORE :READ space:start=RESPR(space)
140 PRINT "Loading Hex...":HEX_LOAD start
150 INPUT "Save to file...";f$
160 SBYTES f$,start,byte:STOP
170 :
180 DEFine FuNction DECIMAL(x)
190 RETurn CODE(h$(x))-48-7*(h$(x)>"9")
200 END DEFine DECIMAL
210 :
220 DEFine PROCedure HEX_LOAD(start)
230 byte=0:checksum=0
240 REPeat load_hex_digits
250    READ h$
260    IF h$="*":EXIT load_hex_digits
270    IF LEN(h$) MOD 2
280       PRINT "Odd number of hex digits in: ";h$
290       STOP
300    END IF
310    FOR b=1 TO LEN(h$) STEP 2
320       hb=DECIMAL(b):lb=DECIMAL(b+1)
330       IF hb<0 OR hb>15 OR lb<0 OR lb>15
340          PRINT "Illegal hex digit in: ";h$:STOP
350       END IF
360       POKE start+byte,16*hb+lb
370       checksum=checksum+16*hb+lb
380       byte=byte+1
390    END FOR b
400 END REPeat load_hex_digits
405 PRINT checksum:STOP
410 READ check
420 IF check<>checksum
430    PRINT "Checksum incorrect. Recheck data. ":STOP
440 END IF
450 PRINT "Checksum correct. Data entered at: ";start
460 END DEFine HEX_LOAD
470 :
480 REMark Space requirements for the machine code
490 DATA 356
500 :
510 DATA "43FA0136": REMark          LEA       CON,A1
520 DATA "347800C6": REMark          MOVEA.W   $C6,A2
530 DATA "4E92": REMark              JSR       (A2)
540 DATA "43FA0138": REMark          LEA       ID,A1
550 DATA "2288":REMark               MOVE.L    A0,(A1)
560 DATA "43FA0132":REMark .PTEXT    LEA       ID,A1
570 DATA "2051":REMark               MOVEA.L   (A1),A0
580 DATA "43FA013C":REMark           LEA       TEXT,A1
590 DATA "347800D0":REMark           MOVEA.W   $D0,A2
600 DATA "4E92":REMark               JSR       (A2)
610 DATA "43FA0122":REMark           LEA       ID,A1
620 DATA "2051":REMark               MOVEA.L   (A1),A0
630 DATA "43FA013C":REMark           LEA       BUFFER,A1
640 DATA "7002":REMark               MOVEQ     #$2,D0
650 DATA "343C0064":REMark           MOVE.W    #BUFLEN,D2
660 DATA "76FF":REMark               MOVEQ     #-1,D3
670 DATA "4E43":REMark               TRAP      #3
680 DATA "5381":REMark               SUBQ.L    #1,D1
690 DATA "41FA012A":REMark           LEA       BUFPOS,A0
```

branch to process a green pixel. Whichever pixel colour we had, the routine will return us to address RET after the pixel has been processed. At this point we ROR the word containing the test bits. This means that next time we loop round, we shall be testing the next bits to the right, which will give us the next pixel to the right. So we are now ready to decrement the bit counter and branch back to the start of the BITLOOP with a DBRA instruction.

After we have looped through all the bits, we leave BITLOOP, and move on to the next word. So we ADD 2 to register A1 to increment the address to the next screen word. We can then decrement the word counter, and loop back to the start of WORDLOOP.

After we have looped through all the words in the row, we leave WORDLOOP, and move on to the next row.

Now processing the pixels involves drawing a 1 by 1 block in each pixel position. When we start a new row, we have to set the coordinates (XORIGIN and YORIGIN) to the start of the next row. This means incrementing YORIGIN for a new row, and making XORIGIN zero for the left of the screen. This done, we can decrement the row counter, and loop back to the start of ROWLOOP.

Tidying Up

After we have looped through all the rows (processed the entire screen), we have some tidying up to to. First, if we are going to use the parameter block again to process another screen, we need to reset XORIGIN and YORIGIN to the top left corner of the screen. Finally, we have to close the channels, before returning to SuperBasic.

Apart from the lines altering XORIGIN and YORIGIN (which are specific to this particular program), what we have covered so far is a method of analysing every pixel on the screen, and then branching to an appropriate routine according to what colour the pixel we find. What those routines will be will depend on what you intend to do with the information. As I mentioned before, the classic situation is using the information to a printer screen dump routine, where appropriate codes are sent to the printer to make a printout of your screen. This requires a good understanding of your printer manual, and varies greatly from printer to printer.

Other applications may include producing a compressed screen file, or simply counting how many pixels are a particular colour, or finding out exactly where a particular block of colour starts and ends. You could turn the picture upside down or back to front, or, with a bit more difficulty, put it on its side. To be completely silly, you could make each colour play a different sound, and turn your picture into music!

But, as promised, we shall turn the picture into monochrome shades of green.

Our routine to colour a block on the screen will alter many registers. In order to deal with this we save all the register on the stack with a MOVEM.L instruction. So this is what the first line does whether we branch

```
700 DATA "3081":REMark                   MOVE.W     D1,(A0)
710 DATA "41FA0124":REMark               LEA        BUFPOS,A0
720 DATA "7601":REMark                   MOVEQ      #1,D3
730 DATA "72FF":REMark                   MOVEQ      #-1,D1
740 DATA "7001":REMark                   MOVEQ      #1,D0
750 DATA "4E42":REMark                   TRAP       #2
760 DATA "4A80":REMark                   TST.L      D0
770 DATA "670A":REMark                   BEQ.S      GOT_FILE
780 DATA "347800CA":REMark               MOVEA.W    $CA,A2
790 DATA "4E92":REMark                   JSR        (A2)
800 DATA "4EFAFFBC":REMark               JMP        PTEXT
810 DATA "43FA00F8":REMark               LEA        FILE,A1
820 DATA "2288":REMark                   MOVE.L     A0,(A1)
830 DATA "243C00008000":REMark           MOVE.L     #$8000,D2
840 DATA "227C00020000":REMark           MOVEA.L    #$20000,A1
850 DATA "76FF":REMark                   MOVEQ      #-1,D3
860 DATA "207A00E4":REMark               MOVEA.L    FILE,A0
870 DATA "7048":REMark                   MOVEQ      #$48,D0
880 DATA "4E43":REMark                   TRAP       #3
890 DATA "227C00020000":REMark           MOVEA.L    #$20000,A1
900 DATA "3A3C00FF":REMark               MOVE.W     #$FF,D5
910 DATA "3C3C003F":REMark .ROWLOOP      MOVE.W     #$3F,D6
920 DATA "3211":REMark     .WORDLOOP     MOVE.W     (A1),D1
930 DATA "7407":REMark                   MOVEQ      #7,D2
940 DATA "363C8080":REMark               MOVE.W     #$8080,D3
950 DATA "3803":REMark     .BITLOOP      MOVE.W     D3,D4
960 DATA "C841":REMark                   AND.W      D1.D4
970 DATA "67000062":REMark               BEQ        BLACK
980 DATA "B843":REMark                   CMP.W      D3,D4
990 DATA "6700007E":REMark               BEQ        WHITE
1000 DATA "3803":REMark                  MOVE.W     D3,D4
1010 DATA "C841":REMark                  AND.W      D1,D4
1020 DATA "0C4400FF":REMark              CMPI.W     #$FF,D4
1030 DATA "6B000066":REMark              BMI        RED
1040 DATA "60000056":REMark              BRA        GREEN
1050 DATA "E25B":REMark    .RET          ROR.W      #1.D3
1060 DATA "51CAFFDE":REMark              DBRA       D2,BITLOOP
1070 DATA "D3FC00000002":REMark          ADDDA.L    #2,A1
1080 DATA "51CEFFCC":REMark              DBRA       D6,WORDLOOP
1090 DATA "47FA0096":REMark              LEA        YORIGIN,A3
1100 DATA "06530001":REMark              ADDI.W     #1,(A3)
1110 DATA "47FA008C":REMark              LEA        XORIGIN,A3
1120 DATA "36BC0000":REMark              MOVE.W     #0,(A3)
1130 DATA "51CDFFB4":REMark              DBRA       D5,ROWLOOP
1140 DATA "47FA0080":REMark              LEA        XORIGIN,A3
1150 DATA "36BC0000":REMark              MOVE.W     #0,(A3)
1160 DATA "47FA007A":REMark              LEA        YORIGIN,A3
1170 DATA "36BC0000":REMark              MOVE.W     #0,(A3)
1180 DATA "43FA0074":REMark              LEA        FILE,A1
1190 DATA "2051":REMark                  MOVEA.L    (A1),A0
1200 DATA "7002":REMark                  MOVEQ      #2,D0
1210 DATA "4E42":REMark                  TRAP       #2
1220 DATA "43FA005E":REMark              LEA        ID,A1
1230 DATA "2051":REMark                  MOVEA.L    (A1),A0
1240 DATA "7002":REMark                  MOVEQ      #2,D0
1250 DATA "4E42":REMark                  TRAP       #2
1260 DATA "4E75":REMark                  RTS
1270 DATA "48E7FFFF":REMark .BLACK       MOVEM.L    A0-A7/D0-D7,-(A7)
1280 DATA "7200":REMark                  MOVEQ      #0,D1
1290 DATA "60000022":REMark              BRA        DRAW
1300 DATA "48E7FFFF":REMark .GREEN       MOVEM.L    A0-A7/D0-D7,-(A7)
1310 DATA "123C0004":REMark              MOVE.B     #4,D1
1320 DATA "60000016":REMark              BRA        DRAW
1330 DATA "48E7FFFF":REMark .RED         MOVEM.L    A0-A7/D0-D7,-(A7)
1340 DATA "123C00E0":REMark              MOVE.B     #$E0,D1
1350 DATA "6000000A":REMark              BRA        DRAW
1360 DATA "48E7FFFF":REMark              MOVEM.L    A0-A7/D0-D7,-(A7)
1370 DATA "123C00D4":REMark              MOVE.B     #$D4,D1
1380 DATA "43FA0028":REMark .DRAW        LEA        ID,A1
1390 DATA "2051":REMark                  MOVEA.L    (A1),A0
1400 DATA "43FA0026":REMark              LEA        BLOCK,A1
1410 DATA "702E":REMark                  MOVEQ      #$2E,D0
1420 DATA "76FF":REMark                  MOVEQ      #-1.D3
1430 DATA "4E43":REMark                  TRAP       #3
1440 DATA "47FA0020":REMark              LEA        XORIGIN,A3
1450 DATA "5253":REMark                  ADDQ.W     #1,(A3)
1460 DATA "4CDFFFFF":REMark              MOVEM.L    (A7)+,D0-D7/A0-A7
1470 DATA "6000FF70":REMark              BRA        RET
1480 DATA "0000":REMark    .CON          DC.W       $0000
1490 DATA "0007":REMark                  DC.W       $0007
1500 DATA "0200":REMark                  DC.W       $0200
1510 DATA "0100":REMark                  DC.W       $0100
1520 DATA "0000":REMark                  DC.W       $0000
1530 DATA "0000":REMark                  DC.W       $0000
1540 DATA "00000000":REMark .ID          DC.L       $00000000
1550 DATA "0001":REMark    .BLOCK        DC.W       $0001
1560 DATA "0001":REMark                  DC.W       $0001
1570 DATA "0000":REMark    .XORIGIN      DC.W       $0000
1580 DATA "0000":REMark    .YORIGIN      DC.W       $0000
1590 DATA "00000000":REMark .FILE        DC.L       $00000000
1600 DATA "000C":REMark    .TEXT         DC.W       $000C
1610 DATA "46494C45204E414D45203F20"
1620 REMark                              DC.B       "FILE NAME ? "
1630 DATA "0000":REMark                  DC.W       $0000
1640 DATA "*",29597
```

to BLACK, GREEN, RED or WHITE. It is not necessary to save all registers, but rather than work out which do and do not need saving, it's simpler to save the lot.

Colouring a block is a TRAP #3 routine. It first requires the colour of the block in register D1. BLACK moves zero into D1 for black, and GREEN moves 4 into D1 for green. RED moves $224 into D1 to give a green/black chequered stipple, and WHITE moves $212 into D1 to give a green/white chequered stipple. The rest is the same whatever colour, so they all branch to common code, DRAW. The console channel ID is required in A0, and the address of the block parameters is required in A1. $2E is required in D0 to tell QDOS that the TRAP #3 routine is going to colour a block of screen. We put -1 in D3 to tell QDOS to keep going until it has finished the routine (infinite timeout). The trap call will then carry out the routine.

After returning from the routine, we need to increment XORIGIN ready for the next pixel to the right. Finally we use a MOVEM.L instruction to restore the registers from the stack, before branching back to RET to process the next pixel.

The whole listing ends with the parameters, etc. The only part that needs any mention is the block definition, four words. The first two are the width and height of the block. Since we are drawing a pixel sized block, these are both 1. The other two are the X and Y origins, and these are set at zero for the top left corner of the screen. These need their own labels, because they are continually altered as pixels are drawn over the entire screen.

Mods to Try

I have mentioned one or two things you can try to modify the listing once you have got it working. Unfortunately, they are mostly concerned with what you might do with the information the routine gathers about pixel colours. None of them have much to do with the heart of the routine, which is looping through the screen pixels to see what colour they actually are. There are other ways of doing this, but you are not likely to discover them from what has been said so far.

One that is worth trying is this. The rotate instructions pushes the bit that falls off the end into the carry bit of the status register. So we could move the word from from screenraminto a data register and then "rotate bits to the left" with a ROL instruction. This would cause the right most bit to go into the carry bit which could then be tested for a 1 with a "branch on carry set" or BCS instruction, or tested for a zero with a "branch on carry clear" or BCC.This method might have the advantage in only requiring a ROL while the original listing requires a ROR followed by an AND to do the same job of isolating a bit for analysis. But it has the disadvantage of only testing one bit at a time, while ROR plus AND can test two, or any number of bits up to 32 at once. We used it to test two bits at a time, one in the even and one in the odd byte. Since the colour depends on two bits, the extra code needed might outweigh any advantage. Anyway, if you feel confident, you might like to give it a try.

As in part 7, I have loaded data into registers for vectored subroutines and trap calls without checking to see if the required data is already there. You will almost certainly be able to reduce the length slightly if you look for redundant lines of code and remove them.

Listing Two is a short SuperBasic program to load and run the code.

Once again, **Listing Three** is Marcus and Simon's Hex Loader for those who do not have an assembler.

Happy coding!

access CLUB

AUSTRIA - DER Computer Club. Contact: Peter Postl, Stiegerg 5, 1150 Wien, Austria.

BELGIUM - Club Sinclair BruQsl. Contact: Jaques Tasset, Aarlenstraat 104, 1040 Brussels, Belgium.

QL Club. Contact: Leon Thianche, Rue Paul Wemaere 12-14, 1150 Bruxelles, Belgium.

FRANCE - QL Contact France. Contact: Jean-Louis Dianoux, 22 Avenue Lenine, 93230 Romainville, France.

GERMANY - Sinclair QL User Club eV. Contact: Franz Herrmann, Talstrasse 21, d-W5460 Ochenfels, West Germany. Magazine: Quasar.

GREECE - QL Athens Club. Contact: Stathis Grigoriadis, Tarsu 6-8, 10440 Athens, Greece.

HOLLAND - Sin_QL_Air. Contact: Secretary: Marco Holmer, J P Coenstraat, 3531 EN Utrecht, Netherlands. Magazine: Quasar. Editor: Same as Secretary. (Chairman: Cor Biemans, Elzenstraat 5, 5461 CL Vehgel, Netherlands.

ITALY - Qitaly Club. Contact: Roberto Orlandi, Via Brescia 26, 25039 Travegliianto (BS), Italy. Tel. +39 30 6863311. Magazine: Qitaly Magazine. Editor: Dr Eros Forenzi, Via Valeriana 44, 23010 Berbenno (SO), Italy. Tel. +39 342 590450.

NORWAY - Norwegian All Sinclair Association (NASA). Contact: P Monstad, NASA, N-5580 Oelen, Norway. Magazine: Sinclair Magazine.

SCOTLAND - Scottish QL Users Group. Contact: Alan Pemberton, 65 Lingerwood Road, Newtongrange, Midlothian EH22 4QQ. Newsletter.

SPAIN - Qliper. Editor: Marcos Cruz, Acacias 44, (Monteclaro), E-28223, Pozuelo de Alarcon, Spain. Mag: Qliper.

SWEDEN - Svenska QL-Gruppen. Contact: Johan Boman, Toftaasgatan 73, 42147 Vastra Frolunda, Sweden. Magazine: QL-Bladet.

Jamten-TCL (International QL echomail conferences) BBS +46-63-133330. Michael Cronsten, Radhusgatan 61E, 83137 Ostersund, Sweden.

TURKEY - QL Qlub. Contact: Bulent Artuz, Prof. Sitesi B/1 D/5, Etiler 80600, Istanbul, Turkey.

UK - Quanta (UK): Membership Secretary, Bill Newell, 213 Manor Road, Benfleet, Essex SS7 4JD. Magazine: Quanta. Editor: Bill Fuggle, 20 Widnes Avenue, Selly Oak, Birmingham B29 6QE. Quanta.

Ashstead sub-group: Derek Stewart, 13 Beales Road, Great Bookham, Surrey KT23 4NA.

Bristol sub-group: Roy Brereton, 94 Teignmouth Road, Clevedon, Avon BS21 6DR.

East Anglia: Geraint Jones, 1 Bridgate Court, Thetford IP24 3AJ. (Also market Rom Switcher.)

Essex sub-group: Dave Walker, 22 Kempton's Mead, Potters Bar, Herts EM6 3HZ.

London sub-group: Jeremy Davis, 6 Elmcroft Crescent, Harrow, Middx. HA2 6HN.

North-east: Derek Stewart, 20 Emily Street, Gateshead.

Northern Ireland: Billy Turkington, Fairyhill, Rostrevor, Newry, Co. Down, BT34 3BB.

Mid-Southern sub-group: Geoff Fish, 44 Billing Avenue, Wokingham, Berks RG11 4JE.

QL User Group (West Midlands): Mike Bedford-White, 16 Westfield Road, Acocks Green, Birmingham B27 7TL.

QL MUG (Merseyside): G Reynolds 051 932 1484, evenings. Meet fortnightly.

Solent sub-group: Graham Evans, 32 Reeves Way, Lowford, Bursledon, Southampton. Tel. 0703 403350.

USA - New England Sinclair QL User Group. Contact: Sherm Waterman, POBox 8763, Boston MA 02114, USA. Mag: NESQLUG News. Editor: Peter Hale, 195 Central Ave., Chelsea, MA 02150, USA.

DJC Dilwyn Jones Computing

41 BRO EMRYS, TAL-Y-BONT, BANGOR,
GWYNEDD, LL57 3YT, GREAT BRITAIN
FAX/TEL: (0248) 354023

DATABASE SOFTWARE

COCKTAILS WAITER	£10 00
DATA DESIGN 3	£60.00
DATA DESIGN API	£20.00
DBEASY	£15.00
DBPROGS	£15.00
FLASHBACK	£25.00
FLASHBACK SE	£40.00
SUPER DISK INDEXER	£12.00

DTP & CLIPART

PAGE DESIGNER 3	£40.00
Available at last!	
UPGRADE FROM PD2	£25.00
SCANNED CLIPART 1 (U)	£10.00
SCANNED CLIPART 2 (U)	£10.00
THE CLIPART (U)	£12.00

FILE HANDLING

FILEMASTER	£12.00
FILES 2 (U)	£12.00
4MATTER & LOCKSMITHE	£23.50
LOCKSMITHE (U)	£14.95
MDV TOOLCHEST	£14.95
THE GOPHER (U)	£12.00
WINBACK 2	£25.00

FILE TRANSFER

CONVERT-PCX	£10.00
DISCOVER	£20.00
MULTI-DISCOVER	£30.00
OPD INTERCHANGE	£15.00
QL-PC FILESERVER	£24.50
SToQL	£24.50
TEXTIDY	£15.00

GAMES & LEISURE

FIVE GAME PACK (U)	£12.50
FLEET TACTICAL COMMAND (U)	£39.95
FTC DATA PRINTER	£9.95
FLIGHTDECK (U)	£15.00
FUGITIVE	£9.95
GREYWOLF	£12.50
OPEN GOLF	£12.50
QUESTION MASTER (U)	£10.00
QUIZ MASTER 2 (U)	£12.50
RETURN TO EDEN	£17.50
SOLITUDE (U)	£15.00
SQUIDGY ROUND THE WORLD (U)	£12.50

LABELLING

ADDRESS BOOK&LABEL PRINTER	£15.00
SUPER DISK LABELLER	£10.00

GENEALOGY

GENEALOGIST 3	£60.00
new pointer driven version	
Upgrade 2nd Edition	£33.00
upgrade other versions, ask	
GENEALOGIST 2ND EDITION	£30.00
BUDGET 128K GENEALOGIST (U)	£12.00

GRAPHICS

IMAGE-D (U)	£10.00
IMAGE PROCESSOR	£15.00
LINE DESIGN	£100.00
PAINTER	£25.00
PICTUREMASTER	£15.00
PICTUREMASTER PLUS	£20.00
QRACTAL	£20.00
QUICK MANDELBROT 3 (U)	£15.00
SCREEN COMPRESSION	£10.00
SCREEN SNATCHER (U)	£10.00
SIMPLE VIDEO TITLES	£5.00
TRANS24 (U)	£10.00

HARDWARE

MICRO PROCESS CONTROLLER	
new redesigned version	£64.50
MPC SOFTWARE	£9.95
NETWORK PROVER	£4.00
QPOWER REGULATOR	£24.95
(unsuitable Gold Card)	
SERMOUSE	£40.00
VOICE ANALYSER-New!	
Ask for details and price	

Add £2 50 postage for MPC, SERMOUSE AND VOICE ANALYSER

MAGAZINES

QREVIEW	£2.00
QL TECHNICAL REVIEW	£2.00
issues 1-8 available	
QL ADVENTURER'S FORUM	£1.75
Issues 1-9 available	
QL LEISURE REVIEW	£2.00
issues 1-2 available	

(UK prices only shown)

PROGRAMMING

BASIC REPORTER (U)	£10.00
DISA	£29.00
DJTOOLKIT (U)	£10.00
EASYPTR 3 PART 1	£40.50
EASYPTR 3 PART 2	£20.00
EASYPTR 3 PART 3	£20.00
MEGATOOLKIT DISK (U)	£25.00
MEGATOOLKIT EPROM (U)	£40.00
QLIBERATOR 3 36	£50.00
BUDGET QLIBERATOR (U)	£25.00
QLOAD AND QREF (U)	£15.00
S EDIT EDITOR	£20.00

SCREEN DUMPS

SIDEWINDER PLUS	£24.95

SPREADSHEET PRINTING

SIDEWRITER (U)	£15.00
3D TERRAIN	£12.50

SUNDRIES

3 5" DSDD DISKS each	£0.40
3 5" DSHD DISKS each	£0.70
3 5" DISK LABELS (roll 100)	£2.00
3 5" LABELS (printer roll)	£2.50
ADDRESS LABELS (roll 100)	£2.00
MICRODRIVE CARTRIDGES	£2.50
MDV LABELS (roll 100)	£2.00
MOUSE MAT	£2.50
3 5" DISK DIVIDERS (20)	£3.00

Add £0 50 postage for labels or mouse mat if only ordering those, or £2 50 postage for disks or disk box divider sets

TEXT

BANTER BANNERS	£25.00
BIBLE TEXT	£20.00
QTYP2	£29.95
QUICK POSTERS (U)	£10.00
ROB ROY PACK (U)	£10.00
SPELLBOUND	£30.00
SPELLBOUND S E	£50.00
TEXTIDY	£15.00
TEXT 'N' GRAPHIX	£20.00

OTHER SOFTWARE

HOME BUDGET (U)	£20.00
PRINTERMASTER (U)	£20.00
QPAC1	£19.95
QPAC2	£39.95
THE Pointer Environment package!	
QTOP	£29.95
SCREEN DAZZLER	£15.00
SCREEN ECONOMISER (U)	£10.00
SLOWGOLD (U)	£5.00
SPEEDSCREEN (U)	£15.00
SPEEDSCREEN EPROM (U)	£30.00
TASKMASTER	£25.00
VISION MIXER 1	£10.00
VISION MIXER PLUS	£22.50

CALL OR WRITE FOR A FREE COPY OF OUR QL SOFTWARE CATALOGUE DETAILING AROUND 100 QL PRODUCTS

PLEASE NOTE: (U) ABOVE MEANS THAT THE SOFTWARE IS SUITABLE FOR USE ON A 128K QL

TERMS: POSTAGE-software sent post free to UK, overseas add £1 00 per program (maximum £3 00) Floppy disks, SERmouse, etc add £2 50 postage (see above) PAYMENT - IN UK currency (pounds sterling) only, please Valid methods of payment are cheque drawn on UK branch of bank or building society, Eurocheque, Postal Order, cash (send by registered post) or by credit card - Visa, Access, Mastercard and Eurocard accepted Please make cheques etc payable to DILWYN JONES COMPUTING Minimum order value £5 00 (due to bank charges) Goods remain property of Dilwyn Jones Computing until paid for in full Orders normally sent out within a few days of receipt, we will attempt to advise if any delay anticipated due to stock problems Orders can be accepted by telephone if paid for with credit card Orders paid with credit card can only be sent to cardholder's address under card company rules FAX - We now have a Fax machine on our usual number (0248) 354023

VISA Master Card Euro Card